III. Advanced Engineering

A. Communications Systems Research

1. Digital Telemetry and Command: Efficient Estimates of Noise Variance in Block-Coded Telemetry,

J. K. Holmes

a. Introduction. Recently, a simple estimator of the noise variance was presented for biorthogonal block-coded telemetry systems, such as that used on *Mariner* Mars 1969. This estimator, using an order statistic (SPS 37-46, Vol. IV, pp. 242–245; SPS 37-58, Vol. II, pp. 37–39), provides a monitor for the signal-to-noise ratio.

In SPS 37-46, an asymptotic formula was used to evaluate the variance of the estimator. More accurate results (to eight places) were obtained in SPS 37-58 by a direct calculation via the computer. In this article, the single-sample maximum likelihood estimate (MLE) of σ, for a single-sample drawn from the largest of the absolute noise-only correlation values, is derived and compared to the one of SPS 37-58. It is shown that the single-sample estimator of SPS 37-58 is asymptotically a MLE. Also, the equation defining the M-sample MLE is derived and shown to have a solution (estimate) different from the average of the single-sample MLE. Furthermore, it is shown that the estimator of SPS 37-58 has a very high asymptotic efficiency when compared to the M-sample MLE and, therefore, under the existing ground rules, is best suited for the purpose of noise variance estimation.

b. Review. The system for which the noise estimator was developed in SPSs 37-46 and -58 is a biorthogonal block-coded telemetry system. Since the code is biorthogonal, only the upper half of the code dictionary need be correlated with the input, since the lower half will correlate to the negative of the first half. The code word presumed to be present corresponds to the correlator with the largest absolute value, the sign determining to which half of the code dictionary the code word belongs. The estimate of the noise variance considered here is obtained from the second largest absolute correlation value which is with high probability due to noise only, with the largest absolute correlation corresponding to the transmitted signal. This largest absolute correlation can, therefore, be used to estimate the relative signal level.

c. Single-sample MLE. The MLE for the noise variance that is obtained here is for a single sample obtained from the largest noise-only absolute correlation term. For M samples, the estimate suggested in SPSs 37-46 and -58 is the average of the single-sample estimator. If we have available the n-vector

$$(x_1, x_2, \cdots, x_n)$$

of noise-only absolute correlation values [$n + 1 = 32$ in the (32,6) code] which can be rearranged in increasing order to form the order statistic

$$(x_{(1)}, x_{(2)}, \cdots, x_{(n)})$$

then $x_{(n)}$ will be the largest absolute correlation value. Because of the ground rules on the complexity of the estimator, only one of the "n" absolute correlation values can be used to estimate the noise variance.

It has been shown (SPS 37-46) that, if ordering is allowed, the best estimator restricted to one correlation will use $x_{(n)}$, the largest absolute correlation value. Since the samples are absolute values of gaussian random variables, the x_i have a distribution function

$$F(x) = 2\left[\Phi(x) - \frac{1}{2}\right] \tag{1}$$

where

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{(2\pi)^{1/2}\sigma} \exp\left(\frac{-t^2}{2\sigma^2}\right) dt \tag{2}$$

The distribution of the largest is given by

$$F(x_{(n)} \leq x) = [2\Phi(x) - 1]^n, \qquad x \geq 0 \tag{3}$$

To obtain the MLE, we must solve the following equation:

$$\frac{\partial}{\partial \hat{\sigma}} \ln [p(x)] = 0 \tag{4}$$

where $p(x)$ is the density function associated with the distribution $F(x)$. From Eq. (3), we have

$$\ln [p(x)] = \ln (2n) + (n-1) \ln [2\Phi(x) - 1] + \ln [\Phi'(x)] \tag{5}$$

Differentiating and setting this equal to zero produces

$$1 = \left(\frac{x}{\hat{\sigma}}\right)^2 - \frac{n-1}{\Phi(x) - \frac{1}{2}}$$

$$\times \left[\Phi(x) - \int_{-\infty}^{x} \frac{t^2}{\hat{\sigma}^2} \frac{1}{(2\pi)^{1/2}\hat{\sigma}} \exp\left(\frac{-t^2}{2\hat{\sigma}^2}\right) dt\right] \tag{6}$$

Integrating by parts and letting $x/\hat{\sigma} = r$, we have

$$1 = r^2 - \frac{n-1}{\Phi(r) - \frac{1}{2}} \cdot \frac{r}{(2\pi)^{1/2}} \exp\left(-\frac{r^2}{2}\right) \tag{7}$$

Once we have the solution to Eq. (7), the single-sample MLE is given by

$$\hat{\sigma} = \frac{1}{r} x_{(n)} \tag{8}$$

For the M-sample average, we would then use

$$\hat{\sigma} = \frac{1}{Mr} \sum_{i=1}^{M} x_{(n)}^i \tag{9}$$

For $n = 1$, the solution to Eq. (7) is trivial, and our single-sample MLE becomes

$$\hat{\sigma} = x_1 \tag{10}$$

But it is biased, since

$$E(\hat{\sigma}) = \int_0^{\infty} \frac{2x}{(2\pi)^{1/2}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx = \left(\frac{2}{\pi}\sigma\right)^{1/2} \tag{11}$$

The estimate of SPS 37-58 for $n = 1$ is given by

$$\tilde{\sigma} = \left(\frac{\pi}{2}\right)^{1/2} x_1 \tag{12}$$

which is, from Eq. (11), unbiased. In fact, all estimates of SPS 37-58 are unbiased for all n. Next, we show that the MLE and the estimate $\tilde{\sigma}$ of SPS 37-58 are asymptotically (in n) the same.

d. Comparison of estimates for large n. The estimate $\tilde{\sigma}$ is defined by

$$\tilde{\sigma} = \lambda_n x_{(n)} \tag{13}$$

where λ_n is tabulated in SPS 37-58 and is defined by the condition $\lambda_n E(x_{(n)}) = \sigma$; moreover, in SPS 37-46, $\tilde{\sigma}$ was shown to be asymptotic to

$$\tilde{\sigma} = \frac{x_{(n)}}{[2\ln(n)]^{1/2}} \tag{14}$$

Now consider the MLE $\hat{\sigma}$ for large n. Note that as n increases, r must increase in order that Eq. (7) is satisfied. Now, rewrite Eq. (7) as

$$\frac{\ln(r^2-1)}{\frac{r^2}{2}} = \frac{\ln(2\pi)^{1/2}}{\frac{r^2}{2}} + \frac{\ln(n)}{\frac{r^2}{2}} + \frac{\ln(r)}{\frac{r^2}{2}} - 1 + \frac{\ln\left(\frac{n-1}{n}\right)}{\frac{r^2}{2}} \tag{15}$$

We have, therefore, that as $n \to \infty$ (and, therefore, as $r \to \infty$)

$$r = [2\ln(n)]^{1/2} + o(1) \tag{16}$$

Hence,

$$\hat{\sigma} = \frac{x_{(n)}}{[2\ln(n)]^{1/2} + o(1)}$$

Consequently, the two single-sample estimators in consideration are asymptotically equal, and, therefore, we conclude that the estimate proposed in SPS 37-58 is asymptotically a single-sample MLE.

e. MLE for M samples. We now obtain the maximum likelihood equation which defines the MLE for M samples and show that it is not the same as the average of the single-sample MLE. The distribution function of M independent samples of a random variable distributed as the largest absolute noise correlation is given by

$$F(\underline{x}_{(n)}) = \prod_{i=1}^{M} [2\Phi(x^i_{(n)}) - 1]^n,$$

$$\underline{x}_{(n)} = (x^1_{(n)}, x^2_{(n)}, \cdots, x^M_{(n)})$$

If we take logarithms and differentiate and set the results equal to zero, we obtain an implicit equation for $\hat{\sigma}$:

$$\hat{\sigma}^2 = \frac{1}{M}\sum_{i=1}^{M}(x^i_{(n)})^2 - \frac{\hat{\sigma}^2(n-1)}{M}\sum_{i=1}^{M}\frac{\left(\frac{x^i_{(n)}}{\hat{\sigma}}\right)\exp\left[-\frac{(x^i_{(n)})^2}{2\hat{\sigma}^2}\right]}{\left[\Phi\left(\frac{x^i_{(n)}}{\hat{\sigma}}\right) - \frac{1}{2}\right](2\pi)^{1/2}} \tag{17}$$

It is clear that the average of the single-sample estimate (Eq. 9) is not a solution to Eq. (17). In fact, it appears to be very difficult to obtain an explicit expression for $\hat{\sigma}$ from Eq. (17). However, the large-sample variance of this MLE can be found from a result given in Ref. 1 where, under rather general conditions satisfied here, it is shown that for a MLE $\hat{\sigma}$ of a single parameter σ, we have

$$\text{var}(\hat{\sigma}) \cong \left\{-ME\left[\frac{\partial^2}{\partial\sigma^2}\ln p(x \mid \sigma)\right]\right\}^{-1} \tag{18}$$

for large M. The coefficient B_n^2 defined by

$$\sigma^2 B_n^2 = \left\{-E\left[\frac{\partial^2}{\partial\sigma^2}\ln p(x \mid \sigma)\right]\right\}^{-1} \tag{19}$$

has been obtained by use of the SDS 930 computer and is compared in Table 1 with C_n^2 of SPS 37-58. The coefficient C_n^2 is defined by

$$\text{var}(\tilde{\sigma}) = C_n^2\,\sigma^2 \tag{20}$$

As can be seen from Table 1, the asymptotic (in M) efficiency of the estimator proposed in SPS 37-58 is quite high for all n considered. Clearly, any improvement by the use of nonlinear estimates would not be worth the

Table 1. Comparison of numerical results

n	C_n^2	B_n^2	Efficiency, %
3	0.195	0.188	96.4
7	0.0962	0.0961	99.8
15	0.0566	0.0563	99.5
31	0.0372	0.0360	96.8
63	0.0259	0.0245	94.6

added complexity. Since for very large n it has been shown that both estimators are identical, it follows that

$$\lim_{n\to\infty}\left(\frac{B_n^2}{C_n^2}\right)=1 \tag{21}$$

It is easy to show that the unrestricted MLE uses all the noise samples available and is given by

$$\hat{\sigma}_{UML}=\left[\frac{1}{Mn}\sum_{i=1}^{n}\sum_{i=1}^{M}(x_j^i)^2\right]^{1/2} \tag{22}$$

And its associated large-sample variance (asymptotic in M) is given by

$$\text{var}\,(\hat{\sigma}_{UML})\cong\frac{\sigma^2}{2Mn} \tag{23}$$

It is not hard to show that for large, fixed M that

$$\lim_{n\to\infty}\left[\frac{\text{var}\,(\hat{\sigma}_{UML})}{\text{var}\,(\bar{\sigma})+\text{var}\,(\tilde{\sigma})}\right]=0$$

since both the variance of $\hat{\sigma}$ and $\tilde{\sigma}$ decrease as $[\ln(n)]^{-2}$ for large n, whereas the variance of $\hat{\sigma}_{UML}$ decreases as n^{-1}. Therefore, MLEs based on the largest noise-only samples are asymptotically (in n) inefficient compared to unrestricted maximum likelihood samples, but require more equipment complexity. However, for the (32,6) code the unrestricted MLE is only about 3 dB better than the restricted one proposed in SPSs 37-46 and -58.

Reference

1. Mood, A. M., *Introduction to the Theory of Statistics.* McGraw-Hill Book Co., Inc., New York, N.Y., 1950.

2. Digital Telemetry and Command: Quantization Requirements for the Multimission Telemetry System,

J. K. Holmes

a. Introduction. Quantization requirements for the data channel quantizers (Fig. 1) are considered here for the multimission telemetry system (MMTS). The requirements are determined for the following constraints:

(1) Degradation of performance in the block-coded mode.

(2) Degradation of performance in the uncoded mode.

(3) Degradation of the accuracy of the signal-to-noise ratio (SNR) estimate in the block-coded mode.

(4) Degradation of the accuracy of the SNR estimates in the uncoded mode.

(5) Dynamic range considerations.

This article considers biorthogonal codes which have been adopted in the MMTS to provide a block-code capability. Both a (32,6) and (16,5) code are provided. The receiver is composed of M ($M = 32$ or 16) correlators, even though there are $2M$ code words. This reduction by a factor of two is possible since the upper half of the code words is a complement of the lower half. If the receiver waveform is designated by $r(t)$, then the correlation values r_i are given by

$$r_i=\int_0^{MT_s} r(t)\,W_i(t)\,dt, \qquad i=1,\cdots,M \tag{1}$$

where T_s is the code symbol time and $W_i(t)$ is the ith code word. Hence, if r_j is the r_i of largest absolute value, $W_j(t)$ is chosen as the transmitted word if $r_j > 0$; otherwise, $-W_j$ is chosen. After each symbol time, the symbol integrator values are quantized and used to form the correlation with the M-stored code words.

In the uncoded mode, since the bits are decoded directly, there is no degradation in bit error probability. However, quantization increases the variance of the SNR estimate, and thereby degrades it. The accuracy of SNR estimates in the coded case is degraded by quantization as in the uncoded case.

Dynamic range variations can be compensated by increasing the quantizer range and number of levels simultaneously so that the quantum levels remain at the same spacing.

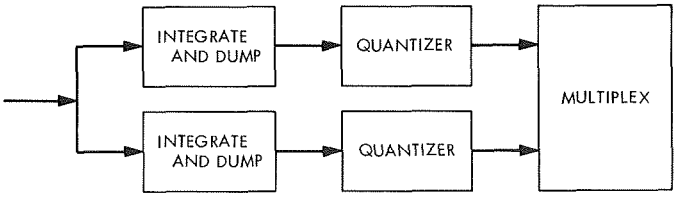


Fig. 1. Data channel

b. Quantizer model. To model the effect of the quantization, the quantizer's effect on the input signal plus noise must be developed. We consider the following model (SPS 37-45, Vol. IV, pp. 290–296) as illustrated in Fig. 2 for eight levels. Each input quantization level is L volts wide, and each corresponding output is at the center of its respective level, the dots in the figure. Altogether, there are 2^k levels (k bit quantization) so, if DR denotes the dynamic range, then the spacing between levels is given by

$$L = \frac{DR}{2^k} \text{ volts} \tag{2}$$

The dynamic range is the range of input voltages that can be quantized with a maximum quantization error of $L/2$ volts. A voltage exceeding the dynamic range is truncated to the last level with an associated increase in quantization error. The basic assumption made here is that the distribution, given that the input voltage falls in a given level (of range L volts), is uniform in that level. The accuracy of this characterization would increase as the level size becomes smaller. Also, at the extremes of the quantizer the error probability density function is actually a section of a gaussian density. However, if the quantization is fine enough, the uniform density will still be a valid assumption. Hence, with the assumption of a uniform density in each level, we have that the mean square of the quantization error at each level is given by

$$\sigma_q^2 = \frac{L^2}{12} \tag{3}$$

Since the range of the quantizer must be large enough to quantize the sum of the signal plus noise, one-half of the dynamic range is taken equal to the integrated signal

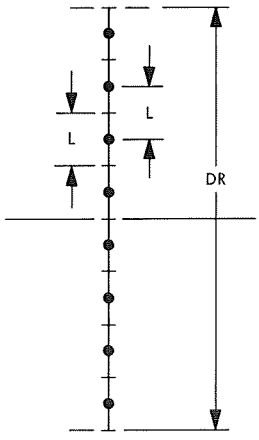


Fig. 2. Quantization scheme for eight levels

symbol AT_s plus three times the standard deviation σ of the integrated noise (SPS 37-45). Any larger samples are truncated, but they only occur with low probability and are assumed to have little effect. Therefore, the level size is given by

$$L = \frac{AT_s + 3\sigma}{2^{k-1}} \tag{4}$$

since $DR = 2(AT_s + 3\sigma)$.

c. Word signal plus noise. We assume that a signal of amplitude A plus white gaussian noise of spectral density $N_0/2$ is the input to the integrator over the symbol time T_s. The output of the integrator, assuming perfect synchronization, is

$$AT_s + N \tag{5}$$

where

$$N = \int_0^{T_s} N(t)\, dt \tag{6}$$

The variance of N is given by

$$\text{var}\,(N) = \frac{N_0}{2} T_s \tag{7}$$

and, hence, the output symbol SNR, without quantization, is given by

$$\text{SNR} = \frac{(AT_s)^2}{\frac{N_0}{2} T_s} = \frac{2E_s}{N_0} \tag{8}$$

where E_s is the symbol energy and is equal to A^2T_s.

Now with quantization after the integration, more noise is added to the correlation with every symbol. Since in the coded mode 16 or 32 samples, each having uniformly distributed random errors, are added up, we may assume that the effect of the quantization is to add gaussian noise of variance $M\sigma_q^2$ to the word correlator outputs. This follows from the central limit theorem.

The total variance becomes

$$M\left(\frac{N_0}{2} T_s + \sigma_q^2\right) \tag{9}$$

The new word signal-to-noise ratio, say SNR′, with the quantization effects included, is given by

$$\text{SNR}' = \frac{M(AT_s)^2}{\frac{N_0}{2}T_s + \sigma_q^2} \tag{10}$$

If we allow a maximum degradation to the SNR of D dB, then we require

$$10\log\left[\frac{M(AT_s)^2}{\frac{N_0}{2}T_s}\right] \leq 10\log\left[\frac{M(AT_s)^2}{\frac{N_0}{2}T_s + \sigma_q^2}\right] + D \tag{11}$$

or

$$\sigma_q^2 \leq \frac{N_0}{2}T_s(10^{D/10} - 1) \tag{12}$$

So, if $D = 0.1$ dB, then we have

$$\sigma_q^2 \leq 0.023\frac{N_0}{2}T_s \tag{13}$$

d. Quantizer equation. Now we can write the quantizer noise variance from Eq. (3) as

$$\sigma_q^2 = \frac{\left[AT_s + 3\left(\frac{N_0}{2}T_s\right)^{1/2}\right]^2}{12(2^{k-1})^2} \tag{14}$$

At this point, it is assumed that AT_s and σ are known exactly so that the dynamic range is known exactly. Since they are actually unknown, extra quantizer bits must be added to compensate for their variation. Using Eq. (14) in Eq. (12) produces

$$\frac{\left[AT_s + 3\left(\frac{N_0}{2}T_s\right)^{1/2}\right]^2}{12(2^{k-1})^2} \leq (10^{D/10} - 1)\frac{N_0}{2}T_s \tag{15}$$

This inequality yields a bound on the required number of bits of quantization. Solving, using Eq. (8), we have, letting $\rho = E_s/N_0$,

$$2^{2k} \geq \frac{2\rho\left[1 + 3\left(\frac{1}{2\rho}\right)^{1/2}\right]^2}{3 \cdot 10^D - 1} \tag{16}$$

which is the quantizer equation relating the required number of bits k versus ρ and the allowable degradation D dB.

e. Requirements at threshold for the (32,6) and (16,5) codes. At threshold, which corresponds to a probability of word error equal to 10^{-2}, the allowable degradation is 0.1 dB. For this error probability, for the (32,6) code, we have, denoting E_B as the bit energy,

$$\frac{E_B}{N_0} = 2.9 \text{ dB}$$

so that the symbol energy per N_0 is given by

$$\frac{E_s}{N_0} = 2.9 \text{ dB} - 10\log_{10}\frac{32}{6} = -4.37 \text{ dB} \tag{17}$$

In evaluating the quantizer equation we find that we must have $k \geq 4$ for a maximum degradation of 0.1 dB.

For the (16,5) code, it is found that $E_s/N_0 = -1.64$ dB or again that $k \geq 4$. Hence, at threshold $k = 4$ is sufficient for both the (32,6) and the (16,5) block codes if AT_s and σ are known *a priori*.

f. Requirements at $PE_w = 10^{-5}$ for the (32,6) and (16,5) codes. At $PE_w = 10^{-5}$, $E_s/N_0 = -0.82$ dB, and with an allowable degradation of 0.2 dB for the (32,6) code, we have, from the quantizer equation, that we must have $k \geq 4$ bits. Under the same conditions for the (16,5) code, it is found that we must have $k \geq 4$ bits also.

We summarize the results in Table 2. This table assumes that $2(AT_s + 3\sigma) = DR$ is held constant. In practice, this is, of course, not true and an additional number of bits must be used to take up signal and automatic gain control variations.

Table 2. Quantization requirements

Block code	Allowable degradation, dB	PE_W	Required k
(32,6)	0.1	10^{-2}	4
(16,5)	0.1	10^{-2}	4
(32,6)	0.2	10^{-5}	4
(16,5)	0.2	10^{-5}	4

g. Quantization effect on PE in the uncoded case. In the uncoded case, since decisions are made bit by bit, there is no degradation for any value of quantization as long as $k \geq 1$.

h. Quantization effect on the var ($\widehat{SNR}$) in the coded case.

Absolute magnitude method, random noise sample. This method estimates the square of the mean output and the variance from the word correlation outputs and forms the ratio to estimate the SNR; call it R. The square of the mean is estimated with the signal presumed present (highest correlation), and the noise variance is formed from a random noise-only correlation. Due to the method involved, it is possible with probability $1/M$ that a random selection will not be available and, so in this case, a zero is outputed for the random noise-only correlation. This effect increases the variance by approximately $M/(M-1)$. Its effect is neglected in what follows.

The estimator is of the form (SPS 37-49, Vol. III, pp. 306–311)

$$\widehat{R} = \frac{\left(\frac{1}{n}\sum_{1}^{n} |x_i^s|\right)^2}{\frac{\pi}{2}\left(\frac{1}{n}\sum_{1}^{n} |x_i|\right)^2} \tag{18}$$

where $|x_i^s|$ is the word correlation value with the signal present and $|x_i|$ is the correlation value of a randomly chosen noise-only correlation. To analyze this expression, we may rewrite $\widehat{R}$ in a manner similar to that given in SPS 37-48, Vol. III, pp. 209–212

$$\widehat{R} = \frac{(AT_s)^2\left[1 + \frac{\delta_n}{(AT_s)^2}\right]}{\sigma^2\left[1 + \frac{\gamma_n}{\sigma^2}\right]} \tag{19}$$

where δ_n and γ_n are, respectively, the random parts of the numerator and denominator.

Under the assumption that with probability approaching one

$$\frac{\delta_n}{(AT_s)^2} \ll 1 \text{ and } \frac{\gamma_n}{\sigma^2} \ll 1 \tag{20}$$

we have, to a good approximation, for n large enough

$$\widehat{R} = \frac{(AT_s)^2}{\sigma^2}\left[1 + \frac{\delta_n}{(AT_s)^2} - \frac{\gamma_n}{\sigma^2} + O(\delta_n\gamma_n) + O(\gamma_n^2)\right] \tag{21}$$

Since the numerator and denominator of the estimate are unbiased, we have

$$E(\delta_n) = E(\gamma_n) = 0 \tag{22}$$

The variance of $\widehat{R}$ is given by

$$\text{var}(\widehat{R}) = \frac{(AT_s)^4}{\sigma^4}\left\{\text{var}\left[\frac{\delta_n}{(AT_s)^2}\right] + \text{var}\left(\frac{\gamma_n}{\sigma^2}\right) - 2E\left[\frac{\delta_n}{(AT_s)^2}\,\frac{\gamma_n}{\sigma^2}\right]\right\} \tag{23}$$

After some algebra, we have

$$\text{var}\left[\frac{\delta_n}{(AT_s)^2}\right] = \frac{4\sigma^2}{n(AT_s)^2}, \qquad \text{var}\left(\frac{\gamma_n}{\sigma^2}\right) = \frac{2(\pi-2)}{n}, \qquad 2E\left[\frac{\delta_n\gamma_n}{(AT_s\sigma)^2}\right] = \frac{1}{n}\left[\frac{2\sigma^2}{(AT_s)^2} - (\pi-2)\right] \tag{24}$$

Therefore, the variance is given by

$$\text{var}(\widehat{R}) = \frac{(AT_s)^4}{\sigma^4}\left[\frac{2\sigma^2}{n(AT_s)^2} + \frac{(\pi-2)}{n}\right] \tag{25}$$

and the relative variance is given by

$$\text{var}\left(\frac{\widehat{R}}{R}\right) = \frac{1}{n}\left[\frac{2\sigma^2}{(AT_s)^2} + \pi - 2\right] \tag{26}$$

Hence, the relative accuracy depends on two terms, one proportional to the noise-to-signal ratio and another that is constant. If we write σ^2 (denoting the thermal noise by σ_{th}^2) as

$$\sigma^2 = \sigma_{th}^2 + \sigma_q^2 \tag{27}$$

we see that a degradation of 0.1 dB due to quantization produces an increase in the relative variance of the esti-

mate that is less than 0.1 dB. This degradation in accuracy can be easily removed, by increasing n by a factor of 1.023. At a degradation of 0.2 dB due to quantization, n can thus be increased to only $(1.023)^2 n$ to remove the effects of quantization.

Largest noise-only method. This estimate uses a new denominator of the form (SPS 37-58, Vol. II, pp. 37–39)

$$(\widehat{\sigma})^2 = \left(\lambda_n \frac{1}{n}\sum_1^n x_i\right)^2 \tag{28}$$

where x_i is the largest noise correlation in absolute value (excluding the signal correlation) from the ith received word. This method is described in SPS 37-58 in detail. Since this estimate has been shown to have a much lower variance for the estimate of the noise variance than the estimate used in the preceding paragraphs, the estimate for R using this noise estimate has lower variance than the estimate described in the preceding paragraphs. The saving is 10.75 dB for the (16,5) code and 11.9 dB for the (32,6) code.

i. Quantization effect on var $(\widehat{SNR})$ in the uncoded case. In this case, the estimator (SPS 37–48) uses symbol values directly to estimate the SNR and is of the form

$$\widehat{R}_L = \frac{\frac{1}{2}\left(\frac{1}{n}\sum_1^n |I_i|\right)^2}{\frac{1}{n-1}\sum_1^n \left(|I_i| - \frac{1}{n}\sum_1^n |I_i|\right)^2} \tag{29}$$

which estimates (1/2) SNR. Since this estimate is biased, an improved estimate is used and is of the form

$$\widehat{R}'_L = \widehat{R}_L^{-1}(R_L) \tag{30}$$

where R_L^{-1} is a function which has less bias than $\widehat{R}_L$ and is asymptotically unbiased for large n for all values of R_L. Its variance is given approximately by

$$\text{var}(\widehat{R'}) = \frac{1}{n}(2R_L + 2R_L^2) \tag{31}$$

assuming that the noise is strictly gaussian. This assumption is reasonable since the sum of a gaussian plus a uniform random variable is essentially gaussian as long as the variance of the uniform is much smaller than that of the gaussian. The relative variance is given by

$$\text{var}\left(\frac{\widehat{R}}{R}\right) = \frac{1}{n}\left[\frac{4\sigma^2}{(AT_s)^2} + 2\right] \tag{32}$$

So if the variance σ^2 is composed of both quantization noise and thermal noise (gaussian), and if the quantization noise increases the total noise by 0.1 dB, we see that the increase in the relative variance is less than 0.1 dB.

j. Dynamic range considerations. Once the number of bits of quantization is determined for the case when the dynamic range is known, the variation in dynamic range must be included. However, once this factor F is known, the additional number of bits needed, k_a, can be obtained from the equation

$$\min k_a: 2^{k_a} \geq F \tag{33}$$

k. Conclusions. The number of quantization levels needed for the MMTS has been obtained under the assumption that the dynamic range was set equal to a fixed value of $AT_s + 3\sigma$. Under actual operating conditions, $AT_s + 3\sigma$ will vary by a factor to be determined so that additional bits are needed.

Two error levels were considered for the coded case, the first with the word error probability equal to 10^{-2} and the second with the word error equal to 10^{-5}. The resulting degradation was shown to be negligible in each case if the number of levels derived here is adopted.

3. Digital Telemetry and Command: Maximum Data Rate and Optimum Data Modulation Index, *U. Timor*

a. Introduction. In a coherent communication system using a phase-locked loop to track the carrier, the average error probability P_E is a function of the total loop signal-to-noise ratio ρ_L, the data modulation index I^2, and the data rate. If P_E is specified, then, for each ρ_L, the data modulation index will determine the data rate which yields P_E. Thus, given P_E and ρ_L, the rate is a function of the data modulation index. The purpose of this work is to find the optimum modulation index and the maximum possible data rate which yield a given error probability P_E; this is done for all data-rate-to-loop-bandwidth ratios.

b. Analysis of the system. Consider a coherent communication system using a phase-locked loop with one-sided bandwidth of b_L to track the carrier. If P is the total

received power and I^2 is the data modulation index, the probability density of the phase error ϕ is given by

$$p(\phi) = \frac{\exp[\rho_L(1-I^2)]}{2\pi I_0[\rho_L(1-I^2)]}$$

where $\rho_L \triangleq P/N_+b_L$ is the total loop signal-to-noise ratio and N_+ is the one-sided spectral density of the noise. Let $1/T$ be the data rate and define

$$\delta = \frac{1}{Tb_L}$$

to be the rate-to-loop bandwidth ratio.

Then, for a completely coherent reception (no phase error), the data signal-to-noise ratio is

$$R = \frac{(PI^2)T}{N_+} = \frac{\rho_L I^2}{\delta}$$

However, there is a degradation of the performance due to the phase error, and the actual signal-to-noise ratio is $R\eta$, where $\eta < 1$ is the detection efficiency.

The behavior of η was investigated by Tausworthe (SPS 37-54, Vol. III, pp. 195–201), with these results. For $\delta \ll 1$, the efficiency is given by

$$\eta_0 = \left[\frac{I_1(\alpha)}{I_0(\alpha)}\right]^2$$

where

$$\alpha = \frac{r+1}{r}\rho_L(1-I^2) - \frac{1}{r\sigma^2_{\sin\phi}}$$

$\sigma^2_{\sin\phi}$ is the variance of $\sin\phi$, $r = 4\xi^2$, and ξ is the damping factor of the phase-locked loop.

For $\delta \gg 1$, the efficiency is given by

$$\eta_\infty = \frac{2}{kR}\left(\text{erfc}^{-1}\left\{\int_{-\pi}^{\pi}\text{erfc}\left[\left(\frac{kR}{2}\right)^{1/2}\cos\phi\right]p(\phi)d\phi\right\}\right)^2$$

where

$$\text{erfc}\, x = \frac{1}{(2\pi)^{1/2}}\int_x^\infty e^{-y^2/2}\,dy$$

and

$$k = \begin{cases} 2 & \text{uncoded or antipodal binary signals} \\ 1 & \text{coded (orthogonal/biorthogonal) signals} \end{cases}$$

For intermediate values of δ, the detection efficiency can be expressed as

$$\eta = (1-a)\eta_0 + a\eta_\infty$$

where the interpolation factor a is a monotone increasing function of δ and is given by

$$a = \frac{\delta}{4}\left[1 - \frac{\delta}{8}(1-e^{-8/\delta})\right]$$

Thus, given ρ_L, I^2, and δ, the detection efficiency η can be determined.

c. The maximum rate-to-bandwidth ratio. Let R_{eq} be the required signal-to-noise ratio to achieve an error probability of P_E. The R_{eq} must satisfy

$$R_{eq} = R\eta$$

Substituting the expressions for R and η, we get the following relation between R_{eq}, ρ_L, I^2, and δ:

$$R_{eq} = \frac{\rho_L I^2}{\delta}\eta_0(1-a) + \frac{2a}{k}\left(\text{erfc}^{-1}\left\{\int_{-\pi}^{\pi}\text{erfc}\left[\left(\frac{k\rho_L I^2}{2\delta}\right)^{1/2}\cos\phi\right]p(\phi)d\phi\right\}\right)^2$$

where η_0 and $p(\phi)$ depend on $\rho_L(1-I^2)$, and a is a function of δ.

Let the error probability, or equivalently R_{eq}, be given. Then for each ρ_L, δ is a function of I^2, given implicitly by the above integral equation. In particular, we are interested in finding the maximum of this function $\delta_{\max}(\rho_L, R_{eq})$, and the corresponding data modulation index $I^2_{\text{opt}}(\rho_L, R_{eq})$ which achieves this maximum. For a given total received power, noise density, and loop bandwidth, $\delta_{\max}$ will give us the maximum possible rate at the required error probabilities.

Since the integral equation cannot be solved analytically, we must resort to numerical methods. The following algorithm was used:

(1) R_{eq} and ρ_L are fixed and a coarse search is made, to restrict I^2_{opt} to an interval of length $2h$. We start with an arbitrary value I^2_0 and compute $\delta_0(I^2_0)$ and $\delta_1(I^2_0 + h)$ by solving, for each case, the integral equation. If $\delta_1 > \delta_0$, we compute $\delta_2(I^2_0 + 2h)$, $\cdots$, $\delta_n(I^2_0 + nh)$ until $\delta_n < \delta_{n-1}$. We then have

$$I^2_{opt} \epsilon [I^2_0 + (n-2)\,h,\ I^2_0 + nh]$$

If $\delta_1 < \delta_0$, we compute $\delta_{-1} = \delta(I^2_0 - h)$ and so on, to get a similar interval of length $2h$ containing I^2_{opt}.

(2) We pick some I^2_1 in the above interval and repeat the procedure with $h_1 < h_0$ (e.g., $h_1 = h_0/10$). We then have I^2_{opt} with an accuracy of $\pm h_1$ and

$$\delta_{n-1} \leq \delta_{max} < \delta_{n-1} + \Delta\delta$$

where

$$\Delta\delta = \max[(\delta_{n-1} - \delta_n), (\delta_{n-1} - \delta_{n-2})]$$

We can repeat the procedure with $h_2 < h_1$ until the accuracies of ΔI^2 and $\Delta\delta$ are accepted.

In the example computed, we took $h_0 = 0.1$, $h_1 = 0.01$, and got δ_{max} within 0.5%.

(3) The computations are repeated for other values of ρ_L.

d. Numerical results. When ρ_L is large, $I^2_{opt} \to 1$, since a very small fraction of the total power will be enough to get an almost coherent reception. Similarly, $\eta \to 1$, since the phase error is very small. Therefore, asymptotically we have

$$\delta_{max}(\rho_L) \underset{\rho_L \to \infty}{\longrightarrow} \frac{\rho_L}{R_{eq}}$$

For other values of ρ_L, numerical calculations are necessary. The resulting values of δ_{max} and I^2_{opt} for several values of R_{eq} and a wide range of ρ_L are given in Fig. 3 (uncoded signals) and Fig. 4 (coded, orthogonal/biorthogonal signals). In both cases, the damping factor was taken to be $\xi = 1/2^{1/2}$ (corresponding to $r = 2$).

e. Application. The described results can be used in the design of coherent communication systems. Given

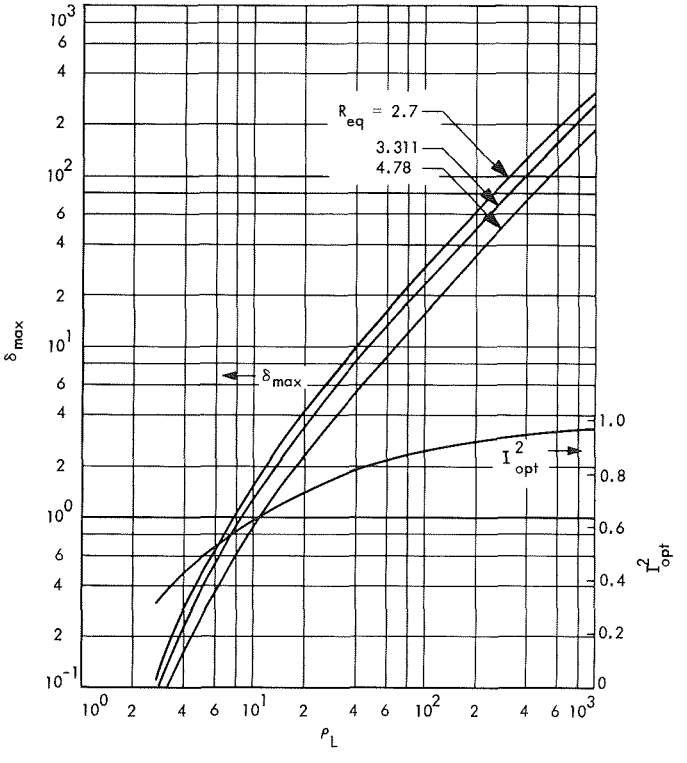


Fig. 3. Maximum rate/bandwidth and optimum data modulation index for uncoded signals

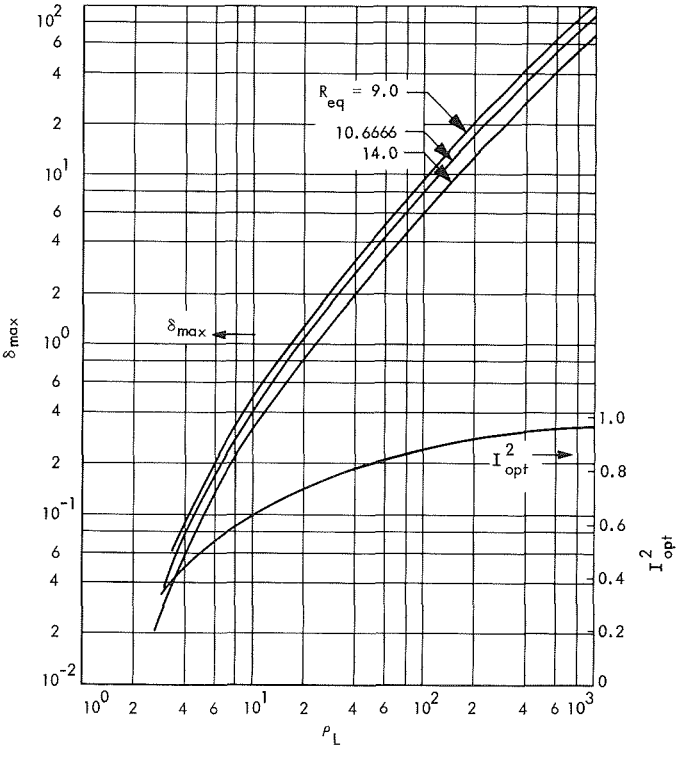


Fig. 4. Maximum rate/bandwidth and optimum data modulation index for coded (orthogonal) signals

the total loop signal-to-noise ratio and the required data signal-to-noise ratio, we can find the maximum possible data rate and the optimum data modulation index. For example, if

$$P = -157.5 \text{ dBmW}$$

$$N_+ = -180.5 \text{ dBmW/Hz}$$

$$b_L = 6 \text{ Hz} = 7.8 \text{ dBHz}$$

then

$$\rho_L = 15.2 \text{ dB} = 33.1$$

If the data are uncoded, then from Fig. 3 we have the following results:

(1) The optimum data modulation index is $I^2 = 0.7$.

(2) The maximum possible rate at $R_{eq} = 3.311$ ($P_E = 5 \cdot 10^{-3}$) is $\delta \cdot b_L = 39.5$ bit/s.

If we want a margin of Δ in the received data signal-to-noise ratio, we have to take the rate from the curve corresponding to $R'_{eq} = R_{eq} + \Delta$.

f. Conclusion. The design curves presented here can be used to select the optimum modulation index for uncoded as well as biorthogonal coherent communication systems over the entire range of possible loop signal-to-noise ratios and data rates. In the past, such design data were available in the form of approximation formulas (SPS 37-44, Vol. IV, pp. 282–290) that were good only at high loop signal-to-noise ratios $\rho_L > 10$ and high rates $\delta \gg 10$, or at very low rates $\delta \ll 10$. The middle range, which covers the *Mariner* Mars 1969 low- and medium-rate systems, had to be designed by trial and error. The graphical data presented here will considerably reduce the design effort for future missions.

4. Digital Devices Development: Screening Test Method for Low-Noise Voltage-Controlled Oscillators,

R. Winkelstein

a. Introduction. A simple straightforward laboratory method has been developed for testing frequency noise in high-stability voltage-controlled oscillators (VCO). Such oscillators are key components in digitally controlled programmed oscillators (SPS 37-36, Vol. III, pp. 54–67) in which control accuracies are desired to better than 10 parts per million. Since the VCO is controlled in a sampled-data feedback system, with a sampling period of 1 s, oscillator frequency stability between sampling times is highly important.

The developed test method makes possible quantitative comparisons between similar VCOs, and has been used to screen out VCOs with unacceptably high noise levels.

b. System block diagram. Figure 5 is a block diagram of the test system. The VCO being tested is the search oscillator of a commercial frequency synthesizer and is referred to as "search oscillator." The output of the search oscillator is mixed with the output of a reference synthesizer and the difference frequency measured by a frequency meter. A dc voltage proportional to the input frequency is generated by the frequency meter and is used as the output test point of the system. This voltage also goes to an integrator which closes the loop by supplying the control voltage to the search oscillator. The

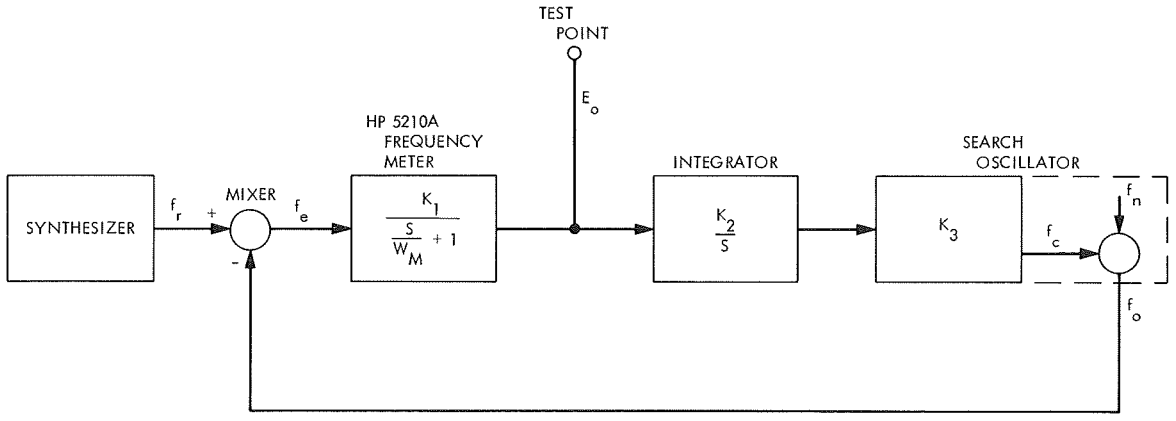


Fig. 5. Test system block diagram

constants of the system allow the loop to control long-term frequency drifts of the search oscillator, but permit short-term frequency deviations to be readily visible at the output test point.

Within each block of interest in Fig. 5 is shown the approximate Laplace transform gain function for the block. The frequency meter is a Hewlett–Packard Model 5210 A with the calibrated offset option. When used in the loop, the meter range switch is set to 1 kHz, the meter scale expanded to "× 10," and the offset to 10. The dc output, taken from the discriminator output jack, is +1 V for a full-scale deflection of 1.1 kHz. Zero voltage output results from an input of 1.0 kHz, and −1 V is obtained when the input is 0.9 kHz. Thus, the gain constant K_1 is 0.01 V/Hz. The "× 10" amplifier within the meter has a frequency cutoff of 10 Hz, which can be approximated by a single pole in the transfer function shown by the term $s/W_M + 1$. The parameter W_M, being 2π times the cutoff frequency, is thus 62.8 rad/s, and s is the complex Laplace transform frequency.

A schematic of the integrator is shown in Fig. 6. It consists of a unity gain inverter followed by an operational integrator. Feedback polarity is set by means of the "feedback" switch. The gain switch, when used in conjunction with the frequency meter range switch, is useful in initially locking the loop. During locked-loop operation, the gain switch is set to the 0.001 position, thus making K_2 equal to 0.001. The 741s are integrated circuit operational amplifiers and the 100-μF capacitor is a subminiature polycarbonate capacitor with greater than 2000-MΩ insulation resistance, made by Component Research Co. Offset currents within the loop are cancelled by the offset control.

The search oscillators tested by the loop were contained in modified Fluke synthesizers Model 644 A. The modification consisted of providing an expanded output of the search oscillator itself, independent of the front panel dial settings. The −1.0 digit on the search oscillator dial produces an output frequency of 1.0 MHz, and +1.0 digit on the dial produces a frequency of 5.0 MHz. For remote control, the −10 to +10 control voltage range could, therefore, produce a frequency swing of 4 MHz. Thus, K_3 is equal to 0.2×10^6 Hz/V. The search oscillator output f_o is considered to be the sum of an ideal controlled frequency f_c and a noise frequency f_n. The purpose of this test method is to measure the effect of f_n.

c. Loop analysis. Multiplying the individual transfer functions, the open loop gain G_o is found to be

$$G_o = \frac{K_1 K_2 K_3 W_M}{S(S + W_M)} = \frac{125.6}{S(S + 62.8)}$$

Since the phase shift is less than 180 deg at the unity gain point of approximately 0.3 Hz, the closed loop is stable. It may be noted that phase shifts in the frequency meter beyond that shown by the approximate transfer function in Fig. 5 caused the loop to become unstable when the loop gain was increased by switching the integrator gain control from 0.001 to 0.01.

Since the purpose of the loop is to measure f_n by monitoring the voltage E_o, the closed-loop transfer function G between f_n and E_o is of prime interest:

$$G = \frac{\text{gain of frequency meter}}{1 + G_o}$$

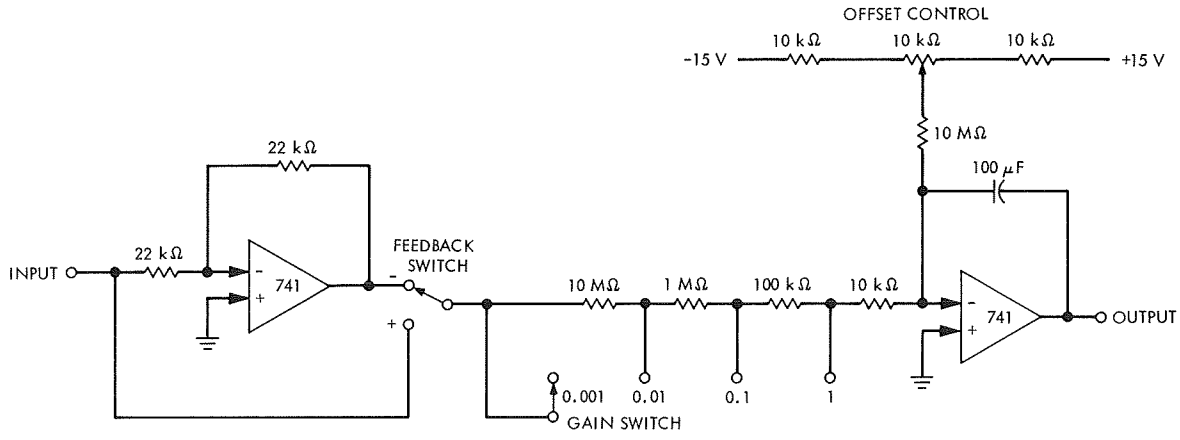


Fig. 6. Integrator

After substitution and algebraic manipulation,

$$G = \frac{S K_1 W_M}{S^2 + S W_M + W_M K_1 K_2 K_3}$$

Factoring the denominator gives

$$G = \frac{S K_1 W_M}{\left[S + \frac{W_M}{2} - \left(\frac{W_M^2}{4} - W_M K_1 K_2 K_3\right)^{1/2}\right]\left[S + \frac{W_M}{2} + \left(\frac{W_M^2}{4} - W_M K_1 K_2 K_3\right)^{1/2}\right]}$$

Since W_M is much larger than $K_1K_2K_3$, the radical may be simplified by approximating it with the first two terms of its binomial expansion

$$\left(\frac{W_M^2}{4} - W_M K_1 K_2 K_3\right)^{1/2} = \frac{W_M}{2} - K_1 K_2 K_3$$

and so

$$G = \frac{SK_1 W_M}{(S + K_1 K_2 K_3)(S + W_M)} = \frac{0.628\,S}{(S + 2)(S + 62.8)}$$

Thus, the transfer function is equivalent to a bandpass filter with corner frequencies at 0.3 and 10 Hz. Gain in the passband is K_1, which is equal to a 0.01-V/Hz deviation.

d. Test results. Figure 7 is a portion of a strip chart recording of a particular test run. The reference synthesizer was set to 1,001,000 Hz, forcing the search oscillator output to be approximately 1 MHz. From the record, it is seen that the maximum peak-to-peak deviation in a 1-s interval can be as high as 50 Hz, although the root-mean-square deviation is considerably less. At search oscillator frequencies of 3 and 5 MHz, observed deviations were reduced by a factor of 2.

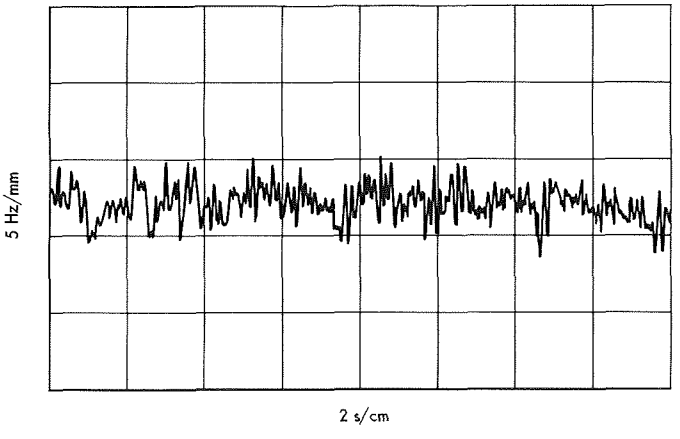


Fig. 7. Test run

This type of performance was acceptable. Of three modified synthesizers tested, the response of the first was as shown in Fig. 7; a second unit had periodic deviations in the order of 200 Hz, and the third unit had deviations in the order of 500 Hz peak-to-peak. Seven more search oscillator VCO units were supplied by the manufacturer and two units whose performance was equivalent to Fig. 7 were substituted in the second and third synthesizers. These test results show that it would be highly desirable to include this test in the test specifications of synthesizers with VCO-type search oscillators.

5. Information Systems: Toward Deep Space Station Automation Software—REGEN, A Binary-to-Symbolic Translator for the SDS 900 Series Languages,

J. W. Layland

a. Introduction. One article in SPS 37-59, Vol. II (pp. 48–54) and two subsequent articles in this volume (pp. 22–31) discuss algorithmic methods for assembling multiprogramming software packages. These algorithms could be applied manually to the symbolic decks of the programs, automatically to the symbolic decks by a program-combining program, or automatically to the binary load modules by a very sophisticated loader. This latter alternative is desirable for conservation of time and storage if several packages utilizing almost the same set of programs are to be generated on demand. This, of course, requires that timing information be supplied via the binary language and that programs must be written which interpret the binary language in greater detail than does the manufacturer-supplied loader.

With relatively few additional programs, the programs necessary to interpret the binary language for a multiprogram scheduler can become a binary-to-symbolic language translator. Because its duties are well defined and its product is more visible, the translator can be written and "debugged" much more easily than could the multiprogram scheduler. Programming of the scheduler when it is eventually done will be an easier task with the important binary language interpreting routines already extant.

The binary-to-symbolic translator is also interesting and useful in its own right for the modification and manipulation of programs for which only the universal binary language version is available, the symbolic version having been lost or being otherwise not available. This article describes REGEN, a binary-to-symbolic translator for the SDS 900 series languages.

b. Program operation. The binary-to-symbolic translation program operates under control of the SDS MONARCH system. All input/output operations are performed by the system library routines MTAPE, CDRP, PYTIO, and PRINT. The input and output devices are selected by a "ΔASSIGN" message to MONARCH prior to loading REGEN. In this assignment message, "BO" specifies the source of the binary program to be translated "X1" specifies a scratch magnetic tape, "SO" specifies the symbolic output device, and "LO" specifies the list output device. REGEN will produce, under breakpoint control, either a symbolic deck which may subsequently be modified and assembled, a code listing whose format closely approximates the format of the listing produced when the input program was initially assembled, or both. If they are set during the initialization for any program, breakpoint 3 deletes the list output and breakpoint 4 deletes the symbolic output. If breakpoint 1 is set, REGEN will pause before translating each program.

Figure 8 shows an example of translator operation. Figure 8a shows the assembly listing of a very short program, Fig. 8b shows an octal dump of the binary tape for this same program, while Fig. 8c shows the listing produced by REGEN from that binary tape. Although it is not nearly as understandable as the original annotated listing, the regenerated listing is far more useful than the octal dump, should the original listing be unavailable. All external labels, both references and definitions, are inserted at the appropriate place in the regenerated deck. Locations which are referenced but do not have an external label are provided with an internal label based upon their address. This internal label consists of an alphabetic–hexadecimal address with $z = 0$, $I = 1, \cdots, Y = 15$, preceded by a prefix which is A9, R9, or C9, depending upon whether that location is absolute, relocatable, or common relocatable. Because the problem of differentiating data from instructions is simplified when all program addresses are relocatable, the output from the translation of a relocatable program will be much more understandable than the output of an absolute program.

c. SDS binary language. The SDS universal binary language can be used to express many more varied descriptions than can be expressed by any of the existing 900-series language processors. Here the basic organization of that language is described. A complete description may be found in the SDS MONARCH or real-time MONITOR manuals (Refs. 1 and 2).

A binary *record* consists of up to 31 words of data. The first word of the record is a control word which specifies the record type, word count, and mode (binary), and contains a parity check on the words of the record. The first three bits of the control word designate the record type: types 000 through 011 are text records, external references and definitions, programmed operator references and definitions, and END records, respectively; type 101 is a data statement record for FORTRAN IV.

The second word of a text record contains the address at which this text group is to be written, two one-bit flags which specify whether this address is absolute, relocatable, or common relocatable, and four one-bit flags which signify the presence of address modifier words for the text words of the record. This is followed by up to twenty-four text words, and, if specified by the flags in word 2, up to four address modifier words. Each modifier word contains one bit for each text word. This bit is a *one* if the associated word is to be modified for load, common, programmed operator, or special input/output relocation.

Type 001 and type 010 records contain up to ten three-word label items. The first two words of each item contain the eight-character binary-coded decimal name. The third word contains a two-bit subtype designator, and for most subtypes, the address to which that name applies and the address modifier flags. For type 010 records, this third word also contains the programmed operator sequence number.

External references are described by "chaining." The external reference item address is the address in the text of the last reference to that label. The address in that

(a)

```
     00000  0 00 00000          1 $KFILT PZE                    DIGITAL FILTER SUBROUTINE
*    00001  0 53 0 00000        2        SKN     KGAIN
     00002  0 01 0 00004        3        BRU     $+2
     00003  0 51 0 00000        4        BRR     KFILT          DONIT FILTER
*    00004  0 54 0 00000        5        SUB     K1OUT
*    00005  0 64 0 00001        6        MUL     KGAIN
     00006  0 46 00014          7        XAB                    DOUBLEPRECISION INTERMEDIATE
     00007  0 55 0 00015        8        ADD     K2OUT
     00010  0 46 00014          9        XAB
*    00011  0 57 0 00004       10        ADC     K1OUT
*    00012  0 35 0 00011       11        STA     K1OUT
     00013  0 36 0 00015       12        STB     K2OUT
     00014  0 51 0 00000       13        BRR     KFILT
     00015  0 00 00000         14 K2OUT  PZE
                               15        END
     00005                        KGAIN
     00012                        K1OUT
```

(b)

```
10455747    42263143    63606060    40100000
02152475    02100000    00000000    05300000    00100004    05100000    05400000    06400001
04600014    05500015    04600014    05700004    03500011    03600015    05100000    00000000
15274000
10752666    42272131    45606060    20100005    42014664    63606060    20100012
30254754    00100016
```

(c)

```
                            1          R9RG    000000                    REGEN
00000  0 00 00000           2  $KFILT  HLT     000000                    REGEN
00001  0 53 00000           3          SKN     KGAIN                     REGEN
00002  0 01 00004           4          BRU     R9L                       REGEN
00003  0 51 00000           5          BRR     KFILT                     REGEN
00004  0 54 00000           6  R9L     SUB     K1OUT                     REGEN
00005  0 64 00001           7          MUL     KGAIN                     REGEN
00006  0 46 00014           8          RCH     000014                    REGEN
00007  0 55 00015           9          ADD     R9W                       REGEN
00010  0 46 00014          10          RCH     000014                    REGEN
00011  0 57 00004          11          ADC     K1OUT                     REGEN
00012  0 35 00011          12          STA     K1OUT                     REGEN
00013  0 36 00015          13          STB     R9W                       REGEN
00014  0 51 00000          14          BRR     KFILT                     REGEN
00015  0 00 00000          15  R9W     HLT     000000                    REGEN
00016                       16          END                               REGEN
00005  KGAIN
00012  K1OUT
```

Fig. 8. Short example of program translation: (a) assembly listing, (b) binary tape, and (c) REGENerated listing

cell points in turn to an earlier reference to the label, ultimately pointing in the program text to the first reference to the external reference label.

The second word of an END record contains the address of the next available location after the current program. A transfer address word and modifier word follow optionally. The data statement record is not generated by any of the processors under MONARCH.

d. Program organization. REGEN makes three passes through the binary text of a program, as it translates it. During the first pass, the input records are read, checked for conformity to the syntax of the binary language, and all legal records are transferred to the X1 file. Illegal records are identified if possible and their identity added to the list output. Data statement records, should they be encountered, are octally dumped onto the list output during pass 1 and not used again.

Three lists are built during pass 1 for later processing. A list of all addresses which are the arguments of instructions begins at the end of the REGEN program and its input/output handlers, and grows upwards. A list of external definition items begins just below the MONARCH resident in upper memory and grows downwards. Since all external definition records must precede all external reference records, the definition list is closed before any external references are encountered. A list of external references begins at the end of the definition list and

grows downwards. List formats are compatible with the formats of the external definition items in the SDS binary language. Pass 1 is complete whenever the end record is encountered.

If any external references appear during pass 1, their chains must be completely identified before actual translation can begin. During pass 2, the X1 file is read backwards and the reference chains are followed through the text records. Each new location where a reference occurs is inserted into the address portion of the external reference item in its list. This location is also removed from the address list where it was placed during pass 1. Pass 2 is complete whenever all reference chains have been followed to the first reference in each.

Actual translation and output formatting is performed during the third pass. Prior to performing the third pass, the address list is sorted into monotonic order for ease in searching, and a program block list is established which designates the range over which the location counter varies within the program being translated. The block list ensures that all cells which are referenced are assigned a location. During pass 3, the location counter is stepped sequentially through the range of the program while the text records are read from X1. If the location appears on the external definition list, the appropriate external label is inserted in the output format. If it appears in the external reference list, the instruction operand is replaced by the appropriate external label. If it appears in the address list but is not externally defined, a hexadecimal label is manufactured for it as described previously. If it is required by the instruction in the text word, a hexadecimal label is similarly manufactured for the instruction operand.

The final listing output line format closely approximates the listing format for a symbol assembly, consisting of an octal representation of the text word, the card number, and the label, operation mnemonic, and operand for the symbolic card image. REGEN's symbolic output consists of only the symbolic card image portion of the same line. Pass 3 is complete when the end record is again encountered.

REGEN consists of approximately 2200 METASYMBOL cards, and assembles into a program which occupies approximately 3K words, exclusive of the input/output subroutines. Except for the sorting of the address list between pass 2 and pass 3, all operations are limited by input/output speed on the SDS 930.

e. Conclusion. This article describes the operation of a binary-to-symbolic translator for the SDS 900 series languages. Although this effort was initiated merely with the intent of familiarization with the SDS binary language—the ultimate goal is a sophisticated multiprogram scheduling loader—this translator can be an extremely useful tool for the examination and manipulation of programs whenever their symbolic source is not readily available but their binary version is.

References

1. *MONARCH manual,* #90-05-66C. Scientific Data Systems, Santa Monica, Calif., July 1967.
2. *Real-Time MONITOR manual,* #90-11-08C. Scientific Data Systems, Santa Monica, Calif., July 1967.

6. Information Systems: Range-Doppler Display System, *A. I. Zygielbaum*

a. Introduction. Advances in the transmitter and receiver capability of the Goldstone DSN Development Facility prior to the most recent Venus conjunction have necessitated the use of more sophisticated processing than was used on previous planetary range-doppler radar observations. In particular, a real-time display of the data and partial maps were required to permit experimenter intervention in the process of iterative averaging of data from mapping many days. A digital video display system (DVDS) was designed to provide this display.

The display consists of an interface to the SDS 930 computer, a data handler, an oscilloscope interface, and a large-screen oscilloscope. With appropriate software support, the display provides the real-time interaction required by the Venus range-doppler mapping effort.

b. Design considerations. For the display of the Venus radar data, the DVDS had to be capable of displaying an image of 80,000 picture elements flicker-free with a maximum number of grey shades. To present a satisfactory image, human visual response had to be considered.

The human observer detects logarithmic brightness change (Ref. 1). A grey scale is defined by equal decibel graduations. Oscilloscope manufacturers (Tektronix and Hewlett–Packard) claim that eight grey shades can be distinguished on a cathode ray tube (CRT). Experiments by the author indicate, however, that twelve grey shades are discernible, although barely.

The displayed image must be stable and linear. Linearity differences of less than 1% are easily noticeable. Also, any "breathing" or jitter of the image is bothersome because the human eye has reflexive motion sensors.

To prevent flicker, the projected image must be refreshed frequently. A human observer sees flicker if a light source is interrupted at a frequency lower than the critical flicker frequency (CFF). Figure 9 plots CFF with respect to the light-to-dark ratio (LDR) and the image luminance (Ref. 1).

The CFF is an important factor in the selection of a CRT phosphor. As the LDR is increased, the CFF decreases. A CRT with longer persistence will require lower refresh rate. Table 3 lists various phosphors and their corresponding decay and required refresh rates for a variable contrast picture (Refs. 1 and 2). A nearly full bright picture necessitates a higher refresh rate. The P 1 phosphor was chosen for the DVDS. This phosphor

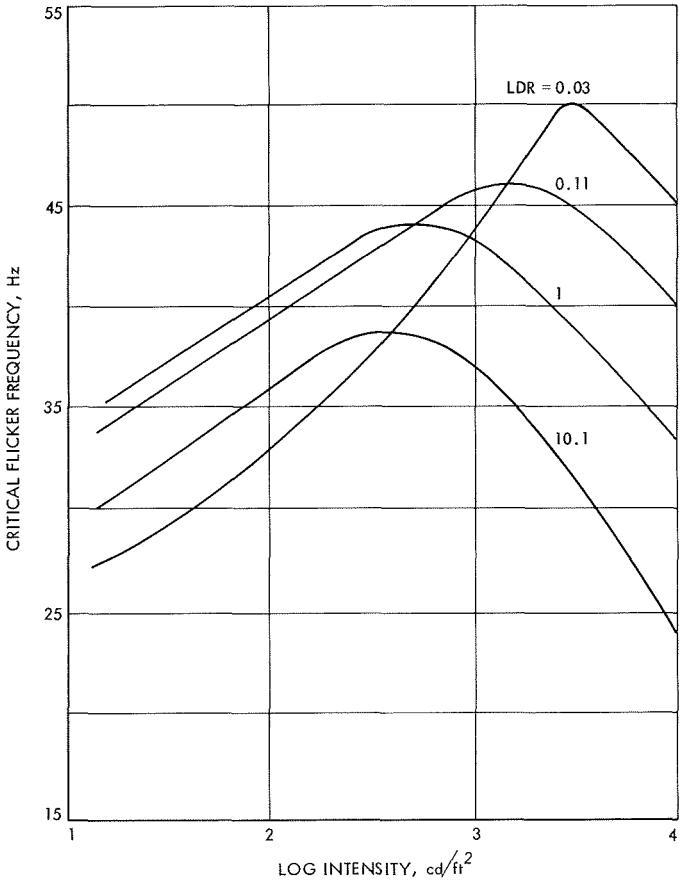


Fig. 9. Human visual response to flicker (taken from Ref. 3)

Table 3. Representative phosphors and their characteristics

Characteristic	P 1	P 4	P 7	P 31	P 39	P 40
Color	Yellow-green	White	Blue	Yellow-green	Yellow-green	White
Refresh rate, Hz	32	40	27	55	25	30
Decay to 0.1%, ms	95	20	1500	32	to 10% 150	to 10% 500

requires a 26-Hz refresh rate at low contrast and approximately 40 Hz at high contrast.

c. System design. The DVDS requires a high data transfer rate. To satisfy this, the data multiplex system (DMS) is used to take data from the SDS 930 computer memory. The DMS can operate at nearly full memory bandwidth and needs only minimal software and hardware interfacing.

The format of the data is a two-dimensional array of points. Each point is designated by position and intensity. This format lends itself to display by a TV-type raster scan.

To make the display as versatile as possible, the system was designed to project a picture of up to 120,000 elements. Since at least 40 frames/s must be presented to eliminate flicker, a picture element rate of 4.8 MHz is required. This requirement, together with the memory transfer bandwidth of 13.7 Mbits, limits each picture element to three bits, i.e., eight grey shades. A four grey shade (2-bit) and an on–off mode are also implemented for flexibility in future use.

d. Data multiplex system. The SDS 930 computer has three well-defined input/output interfaces. Referring to Fig. 10, which is the SDS 930 organization, the available paths to memory include the time-multiplexed communications channel (TMCC), the parallel input/parallel output (PIN/POT) channel, and the DMS. The TMCC is the normal input/output path for peripherals such as magnetic tapes and line printers. PIN/POT channels are used for 24-bit parallel word transfers at slow transfer rates. Both paths are controlled directly by the central processing unit (CPU).

The DMS is a device-controlled second path to memory. Although the CPU must supply word count and

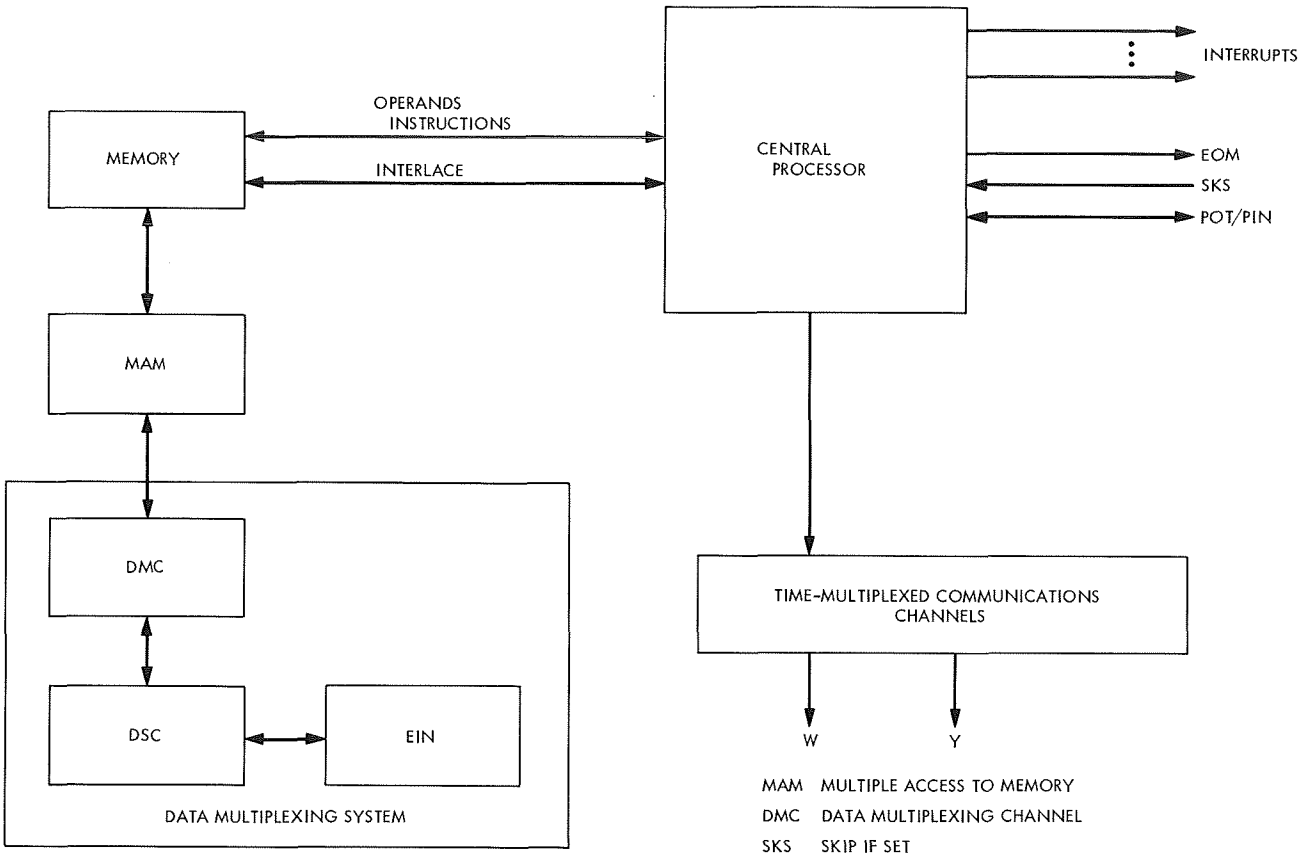


Fig. 10. SDS 930 organization

starting location to the external interlace, an external device can write into, read from, and increment core cells without CPU intervention until word count is exhausted. In the current application, the DMS is used only for readout.

The DMS is controlled through three data lines. Upon activation by the CPU, the DVDS moves these lines to the proper levels for readout under control of the external interlace (EIN). From this point until zero word count, the DVDS requests data from the core memory as needed to fill its buffers.

Two memory locations, even and odd interlace words, are reserved for use by the DMS. The programmer can command the DMS to use the data in either interlace word first and then cycle between the two. Upon activation, the DMS reads out the first interlace word. When the DVDS requests data, the word count is decremented by one and the starting address incremented by one immediately before data is outputted. When the zero word count is reached, an interrupt (ZWC) fires to alert the CPU. The DMS then replaces the exhausted interlace word and reads the other interlace word. Since the DVDS outputs one line per interlace word, the ZWC interrupt indicates end of line.

It is important to note that the DMS has priority over the CPU in accessing any memory location. Therefore, in the accessing and replacing of the interlace words, the DMS steals two cycles from the CPU. This, of course, prohibits operating the DMS at full bandwidth in a bank of memory containing the program because the DMS would lock out the CPU.

e. Data display. Data to be displayed are stored in core as an array of X scan lines. The first word of each line contains the Y-axis location. For example, refer to Fig. 11, picture array segment. The Y axis is controlled by a 9-bit digital-to-analog converter. The Y location is thus randomly selectable.

Data can be presented in three modes: (1) three bits of Z modulation (8 grey shades), (2) two bits (4 shades),

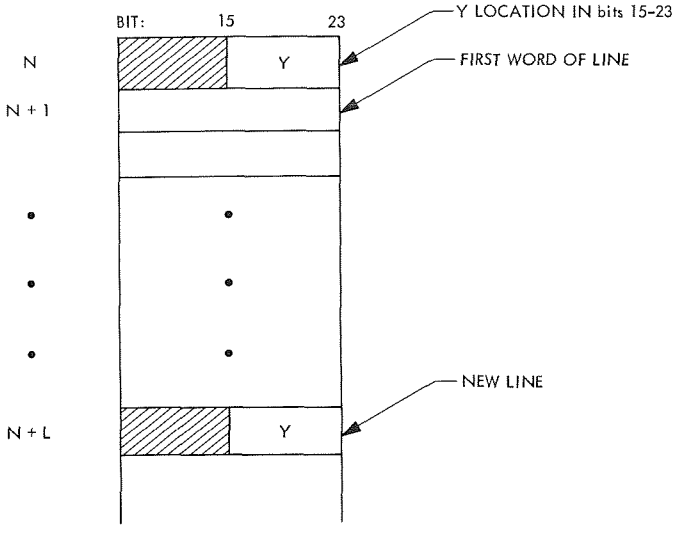


Fig. 11. Picture array segment

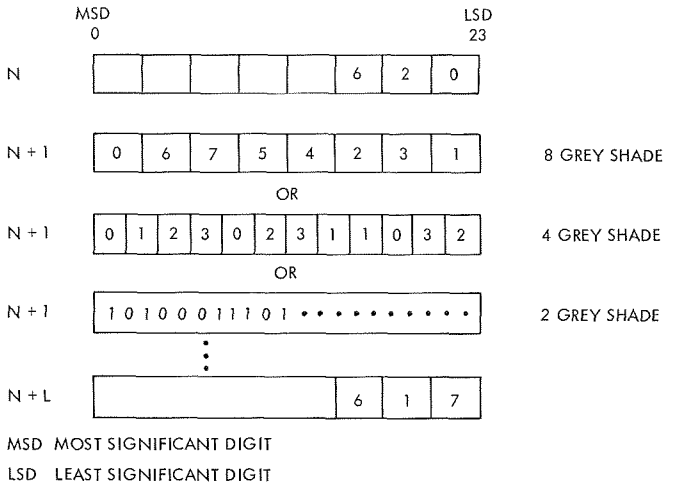


Fig. 12. Picture array organization

or (3) one bit (on–off). Therefore, 8, 12, or 24 picture elements are packed per computer word in the three modes. The data are read out to the display, the most significant digit first. A typical picture array is shown in Fig. 12.

The X axis can be stepped one, two, or four units for every picture element. Between-word blanking, as well as blanking between scan lines, is provided by the Z channel in order to allow the X and Y channels to settle.

Picture elements are presented for an equal time, regardless of the Z mode. Therefore, in the 3-bit mode, data are read out every machine cycle. In the 2-bit mode, data are needed in 2 out of 3 cycles, while only 1 in 3 cycles is accessed in the 1-bit mode.

f. Hardware description. Referring to Fig. 13, DVDS block diagram, the DVDS is divided into three parts. The computer interface control (CIC) commands the DMS to access and output data. It also senses, by means of the ZWC interrupt, that the end of line has been reached, in order to reset the X axis and route the first word of the next line into the Y-axis digital-to-analog buffer.

The mode control is set to its desired state by an EOM–POT sequence. Two octal digits in the POT command designate the Z-axis mode (1, 2, or 3 bits) and the X step (1, 2, or 4 units). The mode control also selects the DMS request rate to once each machine cycle, twice in three cycles or once in three cycles, as explained earlier.

The data control routes data into the Y-axis digital-to-analog buffer or Z-axis shift register. Data (24-bits) are stored in the data acquisition buffer until the Z-axis shift register is emptied. The data control also functions to inform the oscilloscope digital control (ODC) that data is ready for transfer.

The CIC operates at SDS 930 speed and uses the 1.75-μs computer clocks. The ODC operates on an internally generated 15-MHz clock. This clock is turned on only when needed and is turned off after the Z-axis shift register has been emptied.

The ODC operates on a basic cycle of three clock pulses (Fig. 14). The basic controller, the display timing control (DTC), is a three-counter with a fourth state for initialization. The DTC is set to the fourth state at the beginning of the processing of one computer word. This state enables the parallel data transfer from the data acquisition buffer to the Z-axis shift register. The Z register can be shifted one, two, or three steps per cycle depending upon the Z-axis mode selected.

The X axis is stepped once at the beginning of each cycle by adding a known charge to the operational amplifier integrator. At the end of the line, a reset pulse discharges the integrator storage capacitor and sets the beam back to the beginning of the line.

During retrace and X step, the beam is blanked. The Z channel is unblanked at the end of the X step. Because the turn-on time of the Z channel approximately equals the settling time of the X channel, a stable picture element is projected.

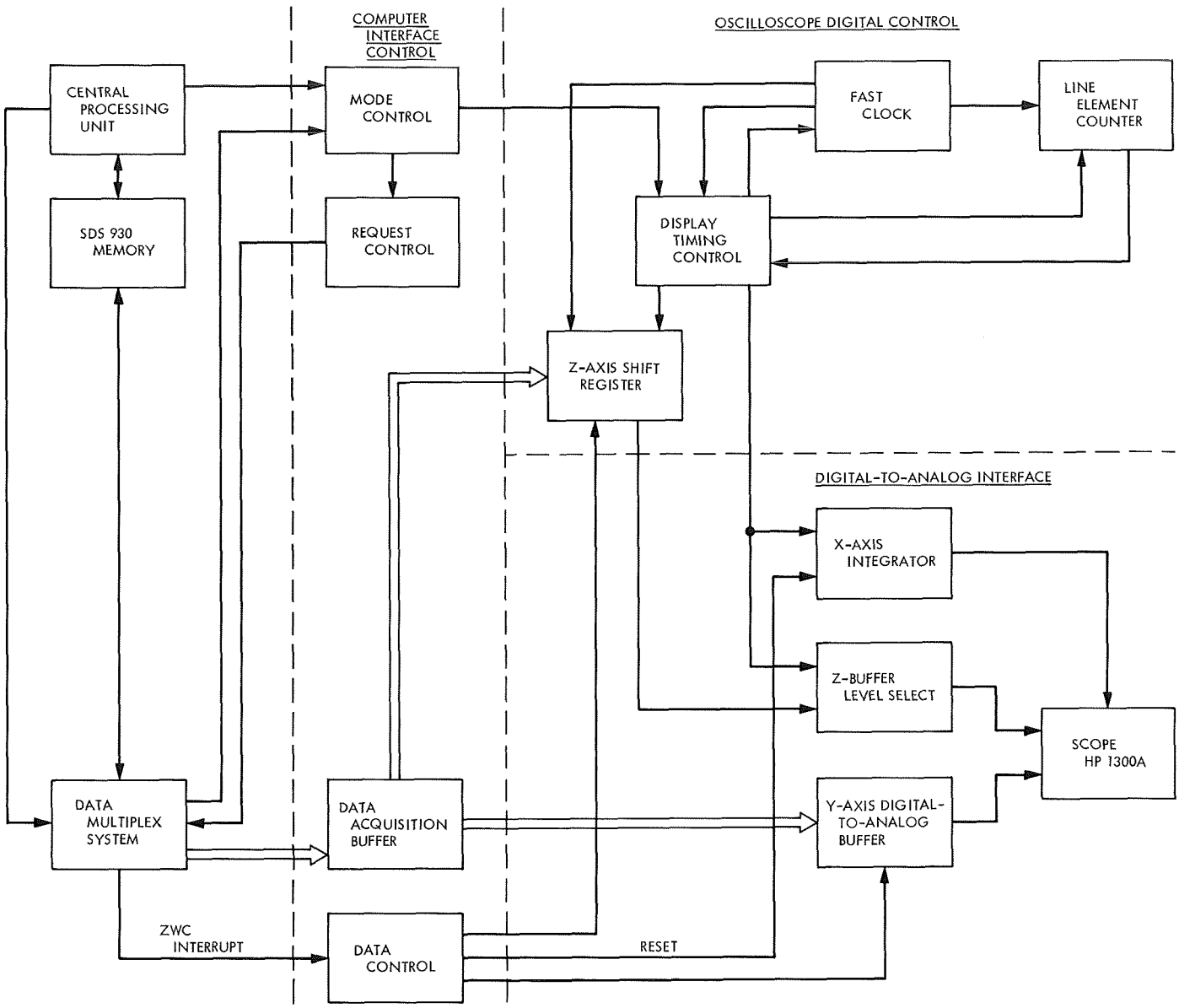


Fig. 13. DVDS block diagram

The line element counter counts the number of Z channel enables. After one word has been processed, the ODC is initialized for the next word.

As stated earlier, the Y-axis word is the first word of the line. After the end of line interrupt, this first word is gated into the Y-axis digital-to-analog buffer. This digital-to-analog converter, a 9-bit device, is allowed 2.25 μs to settle before projection of the first picture element. The Z-channel digital-to-analog converter is an 8-level current select network. It provides one out of eight, one out of four, or on–off levels. A ninth level is provided by the blanking override which has priority over all other levels.

Due to the extremely high speeds required, a 20-MHz oscilloscope—the Hewlett–Packard 1300A—was chosen as the display oscilloscope. The Z-channel amplifier of this oscilloscope has been slightly modified to improve its transient response.

g. DVDS programming. The DMS has an activation sequence consisting of two EOM–POT instructions. The

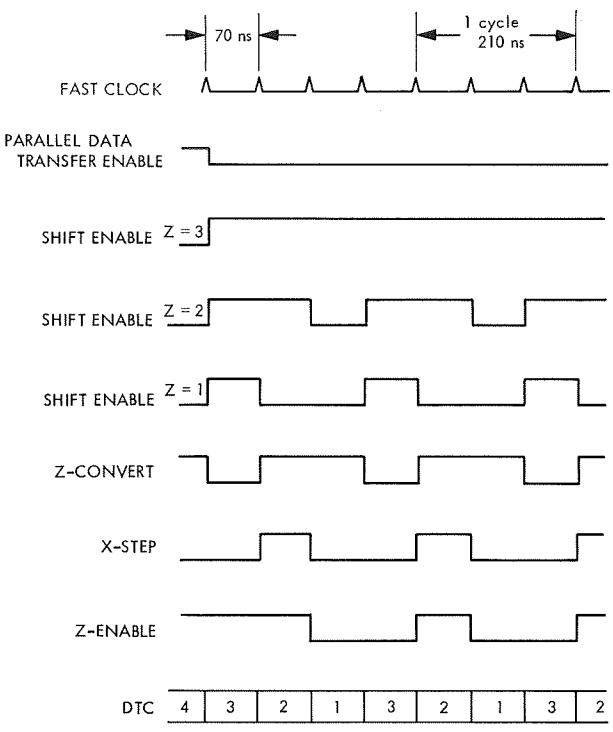


Fig. 14. Display timing

first EOM activates the DMS to control mode. The associated POT controls the end of record or zero word count interrupt arming, interlace word cycling, and designates the first interlace word.

The second EOM sets the DMS to buffer control mode. The lower 12 bits of the succeeding POT'ed word are decoded by devices attached to the DMS. Thus, the *activate* sequence is

EOM	72234	234 is the DSC number
POT	= 6	6 designates ZWC interrupt and cycling
EOM	70234	
POT	= Z X 21	21 is the DVDS designation

The octal digits Z and X have the following meaning:

Value	X	Z
1	step X−4 units	1-bit intensity mode
2	step X−2 units	2-bit intensity mode
3	step X−1 unit	3-bit intensity mode

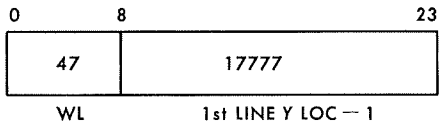
The *deactivate* sequence is simply

EOM	70234
POT	= 0

As indicated previously, at the ZWC interrupt, the last interlace word must be replaced, since it is destroyed by the DMS. The interlace words contain the word count in bits 0–8 and the starting location minus one in bits 9–23. Thus, to output one line, the interlace is loaded with the number of computer words in the line as well as the location of the Y-word minus one.

Table 4. Typical ZWC interrupt routine

LC = number of lines (i.e., 200) less 1
LCA = decremented line count
WL = number of words/line (39)
LIW = last interlace word used
FIW = first interlace word, i.e.,




```
064     BRM     600
600     PZE
        XMA     LIW     save A and LDA with LIW
        SKR     LCA     skip if line count < 0
        BRU     700     to continue frame
        LDA     LC      reset to original
        STA     LCA     word count
        LDA     FIW     reset to FIW
        STA     0234    even interlace word
        STA     0235    odd interlace word
        XMA     LIW     reload A
        BRU     *600
700     XXA             continue frame, put LIW into X
        EAX     18,2    increment X by 18
        STX     0234
        STX     0235
        XXA             LIW to A
        XMA     LIW     reload A
        BRU     *600
```

A typical ZWC interrupt routine for a line length of 304 picture elements in the 3-bit Z-mode (38 computer words plus one Y-word or 39 computer words) is given in Table 4.

h. Applications. The DVDS is used to display planetary radar maps of Venus. Successive iterations can be shown so that convergence of the algorithm can be checked. The DVDS can be used to produce motion pictures or moving displays, such as can be used to examine the behavior of the Viterbi decoder for convolutional codes (SPS 37-54, Vol. III, pp. 171–177). Finally, the system is useful as a very fast X–Y plotter. Current software plots approximately 7000 data points per second into the raster scan line array.

References

1. Luxemberg, H. R., and Kuehn, R. L., *Display Systems Engineering.* McGraw–Hill Book Co., New York, N.Y., 1968.
2. *General Catalog,* p. 5. Tektronix Corp., Beaverton, Oregon, 1969.
3. Bartlett, S., "The Neural Determination of Critical Flicker Frequency," *J. Exp. Psychol.,* Vol. 21, 1937.

7. Information Systems: Scheduling Algorithms for Multiprocessors in a Hard Real-Time Environment,

C. L. Liu

a. Introduction. Scheduling algorithms for a single processor in the hard real-time environment were studied in SPS 37-59, Vol. II, pp. 48–54, and in a companion article in this SPS (pp. 31–37). In this article, we report some results on scheduling algorithms for multiple rather than single processors. Few of the results obtained for a single processor generalize directly to the multiple processor case; bringing in additional processors adds a new dimension to the scheduling problem. The simple fact that a task can use only one processor even when several processors are free at the same time adds a surprising amount of difficulty to the scheduling of multiple processors. As in the single-processor studies, we assume that all tasks to be executed on a priority basis have periodic requests, fixed computation time at each request, and deadlines which correspond to the subsequent request for each task. A discussion of the appropriateness of these assumptions to the DSIF can be found in SPS 37-59, Vol. II, pp. 48–54.

b. Period-driven scheduling algorithm for multiprocessors. Suppose m tasks are to be scheduled on n processors by the period-driven scheduling algorithm. At any instant, then, among those tasks demanding processor time, the n tasks with the shortest periods will be executed. (Of course, if there are less than n tasks demanding processor time, all of these tasks will be executed.) As is expected, the period-driven scheduling algorithm for multiprocessors is not optimum. To see this, we note that for two processors and a set of three tasks with $T_1 = 4$, $C_1 = 3$, $T_2 = 4$, $C_2 = 3$, $T_3 = 5$, and $C_3 = 2.5$, the period-driven scheduling algorithm is not even feasible.

We have not been able to obtain a necessary and sufficient condition on the feasibility of the period-driven scheduling algorithm. The results here provide a sufficiency test on feasibility.

We study first a special case where $n + 1$ tasks are to be scheduled on n processors. Without loss of generality, we assign each of the n tasks with the shortest periods to a dedicated processor. The task with the longest period will run on any of the n processors whenever a processor becomes free. In other words, the task with the longest period will be run as a background task to the n tasks with shorter periods. Given the values of $T_1, T_2, \cdots, T_{n+1}$, $C_1, C_2, \cdots, C_n$, theorem 1 gives a lower bound to the value of C_{n+1} such that the period-driven scheduling algorithm is feasible.

We define two sets of functions:

$$f_j(t) = \begin{cases} 0, & t \leq T_j - \delta_j \\ t - (T_j - \delta_j), & T_j - \delta_j \leq t \leq 2T_j - \delta_j - C_j \\ T_j - C_j, & 2T_j - \delta_j - C_j \leq t \leq 3T_j - C_j - 2\delta_j \\ t - 2(t_j - \delta_j), & 3T_j - C_j - 2\delta_j \leq t \end{cases}$$

$$g_j(t) = \begin{cases} \left(\left[\frac{t}{T_j}\right] - 1\right)\delta_j + f_j\left(\left\{\frac{t}{T_j}\right\} + T_j\right), & t \geq T_j \\ f_j(t), & t \leq T_j \end{cases}$$

for $j = 1, \cdots, n$, where the δ_j's are computed recursively as

$$\delta_1 = T_1 - C_1$$

$$\delta_i = T_i - C_i + \max\left(g_1(C_i), g_2(C_i), \cdots, g_{i-1}(C_i)\right)$$

for $i = 2, \cdots, n$. The notation $[x]$ means the greatest integer less than x and $\{x\} = x - [x]$, the fractional part of x.

Theorem 1. A lower bound δ_{n+1} to the value of C_{n+1} such that the period-driven scheduling algorithm is feasible for $C_{n+1} \leq \delta_{n+1}$ is

$$\delta_{n+1} = \max\left(g_1(T_{n+1}), g_2(T_{n+1}), \cdots, g_n(T_{n+1})\right)$$

Proof. By the *background computation time* on a set of processors, we mean the total amount of non-overlapping processor time on these processors available to a background task. Note that the word "non-overlapping" is important, because even when several processors are free at a certain instant, the background task can make use of only one of these processors.

We show first that δ_i, $i = 1, 2, \cdots, n$, is a lower bound to the background computation time on processors $1, 2, \cdots, i$ within each cycle or task i. We note that within a cycle of task i, the ith processor will be occupied by the ith task for C_i seconds and will then be free for the subsequent $T_i - C_i$ seconds. We now want to estimate the background computation time on processors $1, 2, \cdots, i-1$ during the C_i seconds when the ith processor is occupied by the ith task. We claim that within any C_i contiguous seconds, the background computation time on processors $1, 2, \cdots, j$ is lower-bounded by $g_j(C_i)$. We note that within each cycle of task j there are at least δ_j seconds of background computation time on processors $1, 2, \cdots, j$. Moreover, at least C_j of these δ_j seconds are available at the end of a cycle. Within C_i contiguous seconds, there must be at least $[C_i/T_j] - 1$ complete cycles of task j, as illustrated in Fig. 15. In these $[C_i/T_j] - 1$ complete cycles, the background computation time on processors $1, 2, \cdots, j$ is at least $([C_i/T_j] - 1)\ \delta_j$ seconds. To estimate the background computation time on processors $1, 2, \cdots, j$, within $\Delta_1 + \Delta_2$, we see that Fig. 15 shows the most unfavorable distribution of background computation time in the two cycles at the ends of the C_i seconds. We, thus, have the estimation

$$\text{background processor time within } \Delta_1 + \Delta_2 \geq \begin{cases} 0, & \Delta_1 + \Delta_2 \leq T_j - \delta_j \\ (\Delta_1 + \Delta_2) - (T_j - \delta_j), & T_j - \delta_j \leq \Delta_1 + \Delta_2 \leq 2T_j - \delta_j - C_j \\ T_j - C_j, & 2T_j - \delta_j - C_j \leq \Delta_1 + \Delta_2 \leq 3T_j - C_j - 2\delta_j \\ (\Delta_1 + \Delta_2) - 2(T_j - \delta_j), & 3T_j - C_j - 2\delta_j \leq \Delta_1 + \Delta_2 \end{cases}$$

It follows that $g_j(C_i)$ is a lower bound to the background computation time on processors $1, 2, \cdots, j$ within any C_i contiguous seconds. This proves the theorem.

To illustrate the application of Theorem 1, let us consider the problem of scheduling three tasks with $T_1 = 3$, $C_1 = 2$, $T_2 = 4$, $C_2 = 3$, and $T_3 = 7$ on two processors. We want to estimate the value of C_3 such that the deadline scheduling algorithm is applicable. We have

$$\delta_1 = T_1 - C_1 = 3 - 2 = 1$$

$$\begin{aligned} \delta_2 &= T_2 - C_2 + g_1(C_2) \\ &= 4 - 3 + g_1(3) \\ &= 4 - 3 + 1 = 2 \end{aligned}$$

$$\begin{aligned} \delta_3 &= \max\left(g_1(T_3), g_2(T_3)\right) \\ &= \max\left(g_1(7), g_2(7)\right) \\ &= \max(2, 3) \\ &= 3 \end{aligned}$$

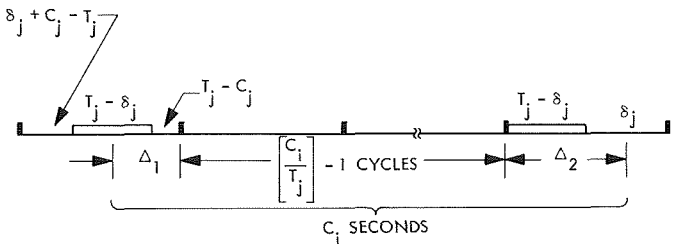


Fig. 15. Background computation time within C_i

As it turns out, the largest possible value of C_3, such that the period-driven scheduling algorithm is feasible, is also 3, so the deadline-driven and period-driven scheduling algorithms are equally efficient for this situation.

It is interesting to compare the lower bound in Theorem 1 and the actual largest possible value of C_n. Some results are tabulated in Table 5.

The results in Theorem 1 can be applied to the general case when m tasks are to be scheduled on n processors. After computing δ_{n+1} according to Theorem 1, we can lower-bound the background computation time on processors $1, 2, \cdots, m$ in every cycle of task $n + 2$ by

$$\delta_{n+2} = \left(\left[\frac{T_n + 2}{T_n + 1}\right] - 1\right)(\delta_{n+1} - C_{n+1})$$

Similarly, we can have

$$\delta_{n+3} = \left(\left[\frac{T_n + 3}{T_n + 2}\right] - 1\right)(\delta_{n+2} - C_{n+2})$$

and so on. As can be expected, such lower bounds deteriorate very rapidly as the number of tasks increases.

c. Deadline-driven scheduling algorithm for multiprocessors. We want to investigate now the deadline scheduling algorithm for multiprocessors. Unfortunately, we have not yet been able to obtain a necessary and sufficient condition on the feasibility of the algorithm. Neither were we able to obtain a sufficient condition similar to Theorem 1. However, there are some interesting observations.

First, we note that the deadline-driven scheduling algorithm is not always optimum, as the following example illustrates. Let $T_1 = 4$, $C_1 = 3$, $T_2 = 4$, $C_2 = 3$, $T_3 = 5$, and $C_3 = 2.5$. The utilization factor will be 100% if these three tasks can be scheduled on two processors. It is easy to see that for this set of tasks the deadline scheduling algorithm is not feasible. However, these three tasks can be scheduled on two processors, as shown in Fig. 16.

Table 5. Comparison of lower bound and actual background computation time for period-driven scheduling of $n + 1$ tasks on n processors

T_i	C_i	T_{n+1}	Lower bound according to theorem 1	Actual maximum
3 4	2 3	7	3	3
3 5	2 4	7	2	2
5 7	2 2	9	5	7
5 7	2 2	31	21	27
5 31	2 7	41	33	34
5 7	4 2	15	10	11
5 20	4 17	30	6	8
5 31	4 27	41	8	11
13 27	9 25	41	12	12
5 7 9	2 2 3	11	7	9
3 5 13 25	2 4 11 21	41	15	21
5 7 9 13 15 31	4 6 7 11 12 27	35	11	18

Second, unlike the single-processor case, the deadline scheduling algorithm is not always superior to the period-driven scheduling algorithm. In other words, there are examples in which the period-driven schedule is feasible while the deadline scheduling algorithm is not. Although

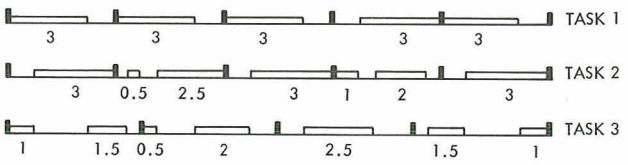


Fig. 16. A feasible schedule for three tasks with 100% utilization of two processors: $T_1 = 4$, $C_1 = 3$, $T_2 = 4$, $C_2 = 3$, $T_3 = 5$, $C_3 = 2.5$

this might sound surprising at first, a moment of reflection indicates that indeed this should have been anticipated. Without complete knowledge of the future timing structure of the tasks to be scheduled, both the period-driven and deadline-driven scheduling algorithms make their decisions on the basis of some local information. Therefore, we should not expect one scheduling algorithm to be always better than another. Table 6 contains some comparison.

In view of the results in Table 6, we conjecture that there is no particular advantage in employing the deadline scheduling algorithm in the case of multiprocessors.

d. Conclusion. Many problems concerning scheduling algorithms for multiprocessors in a hard real-time environment are still open for further investigation, for example, proving or disproving the existence of optimum scheduling algorithms and finding such algorithms if they do exist. One difficulty in designing "good" scheduling algorithms is the problem of predicting the global timing structures of the tasks on the basis of some local information (e.g., the deadlines of the current requests). Both the period-driven scheduling algorithm and the deadline-driven scheduling algorithm offer some very simple decision rules for scheduling a set of tasks on several processors. However, we have not yet obtained any measurement on the effectiveness of the algorithms as compared to other scheduling algorithms.

8. Information Systems: Scheduling Algorithms for Hard Real-Time Multiprogramming of a Single Processor, C. L. Liu

a. Introduction. In SPS 37-59, Vol. II, pp. 48–54, Layland studied an algorithmic method for scheduling tasks with periodic requests on a time-shared computer. In this article, another algorithmic method is proposed for single processors and investigated. This method is optimum in the sense that if a set of tasks can be scheduled by some algorithmic method, it can also be scheduled by this method. In other words, the least upper bound on processor efficiency using this algorithm is uniformly 100%, and this efficiency can be attained by adjusting

Table 6. Comparison of period-driven and deadline-driven scheduling

Number of processors	Number of jobs	T_i	C_i	T_{n+1}	Maximum C_{n+1} using period-driven scheduling algorithm	Maximum C_{n+1} using deadline-driven scheduling algorithm
2	3	4 4	3 3	5	1	2
2	3	3 4	2 3	7	3	3
2	3	5 20	4 17	30	8	9
2	3	5 7	2 2	9	7	7
2	3	5 7	2 2	15	11	11
2	3	5 31	4 27	41	11	9
4	5	3 5 13 25	2 4 11 21	41	21	17

either the period or the computation time of any one of the tasks to be scheduled.

We follow the assumptions discussed in SPS 37-59, Vol. II, on the hard real-time environment. We shall refer to the scheduling algorithm in that article as the *period-driven scheduling algorithm* because priorities are assigned to tasks according to their periods. When a set of tasks are scheduled by some scheduling algorithm, we say that there is an overflow at time t if a request that should have been satisfied by time t was not satisfied at that time. For a given set of tasks, a scheduling algorithm is said to be *feasible* if the tasks are scheduled such that no overflow occurs.

b. Deadline scheduling algorithm. We define a *deadline-driven scheduling algorithm* as one in which priorities are assigned to tasks according to the deadlines of their current requests. A task will be assigned the highest priority if the deadline of its current request is the nearest, and will be assigned the lowest priority if the deadline of its current request is the farthest. At any instant, the task of the highest priority with a yet unfulfilled request will be executed. Such a method of assigning priorities to the tasks is a dynamic one, in contrast to a static assignment in which priorities of tasks do not change with time.

Given a set of tasks with periods $T_1, T_2, \cdots, T_m$ and computation time $C_1, C_2, \cdots, C_m$, we want to establish a necessary and sufficient condition for the feasibility of the deadline-driven scheduling algorithm. We have first a lemma:

Lemma 1. When the deadline-driven scheduling algorithm is used to schedule a set of tasks on a single processor, there is no processor idle-time prior to an overflow.

Proof. Suppose that there are processor idle-periods prior to an overflow. To be specific, starting at time 0, let t_3 denote the time at which an overflow occurs, and let

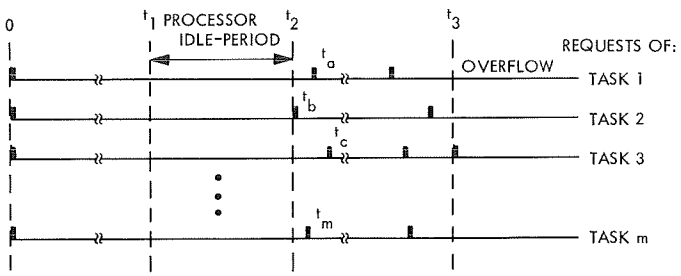


Fig. 17. Processing overflow following a processor idle period

t_1 and t_2 denote the beginning and the end, respectively, of the processor idle-period closest to t_3 (that is, there is no processor idle-time between t_2 and t_3). The situation is illustrated in Fig. 17, where the request-times of the first request of the m tasks after the processor idle-period are denoted $t_a, t_b, \cdots, t_m$.

Suppose that from t_2 on we move all requests of task 1 up so that t_a will coincide with t_2. Since there was no processor idle-time between t_2 and t_3, there will be no processor idle-time after t_a is moved up. Moreover, an overflow will occur either at or before t_3. Repeating the same argument for all other tasks, we conclude that if all tasks are initiated at t_2, there will be an overflow with no processor idle-period prior to it. However, this is a contradiction to the assumption that starting at time 0 there is a processor idle-period prior to an overflow. This proves Lemma 1.

We are now ready to establish the following theorem:

Theorem 1. For a given set of m tasks, the deadline-driven scheduling algorithm is feasible if and only if

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_m}{T_m} \leq 1$$

Proof. To show the necessity, let us calculate the total demand of computation time by all tasks between $t = 0$ and $t = T_1 T_2 \cdots T_m$, which is

$$(T_2 T_3, \cdots, T_m)\, C_1 + (T_1 T_3, \cdots, T_m)\, C_2 + \cdots + (T_1 T_2, \cdots, T_{m-1})\, C_m$$

If the total demand exceeds the available processor time, that is, if

$$(T_2 T_3, \cdots, T_m)\, C_1 + (T_1 T_3, \cdots, T_m)\, C_2 + \cdots + (T_1 T_2, \cdots, T_{m-1})\, C_m > T_1 T_2, \cdots, T_m \qquad (1)$$

there is clearly no feasible scheduling algorithm. Inequality (1) can be written as

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_m}{T_m} > 1$$

To show the sufficiency, let us assume that the condition

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_m}{T_m} \leq 1$$

is satisfied and yet the scheduling algorithm is not feasible. That is, there is an overflow between $t = 0$ and $t = T_1T_2 \cdots T_m$. Moreover, according to Lemma 1, there is a time ($0 \leq T \leq T_1T_2 \cdots T_m$) at which there is an overflow with no processor idle-time between 0 and T. To be specific, let $a_1, a_2, \cdots, b_1, b_2, \cdots$ denote the request-times of the m tasks immediately prior to T, where $a_1, a_2, \cdots$ are the request-times of tasks with deadlines at T, and $b_1, b_2, \cdots$ are the request-times of tasks with deadlines beyond T. This situation is illustrated in Fig. 18.

In the following, we use $[x]$ to denote the greatest integer less than x. We examine two cases:

Case 1. None of the computations requested at $b_1, b_2, \cdots$ is carried out before T. In this case, the total demand of computation time between 0 and T is

$$\left[\frac{T}{T_1}\right]C_1 + \left[\frac{T}{T_2}\right]C_2 + \cdots + \left[\frac{T}{T_m}\right]C_m$$

Since there is no processor idle-period, we have

$$\left[\frac{T}{T_1}\right]C_1 + \left[\frac{T}{T_2}\right]C_2 + \cdots + \left[\frac{T}{T_m}\right]C_m > T$$

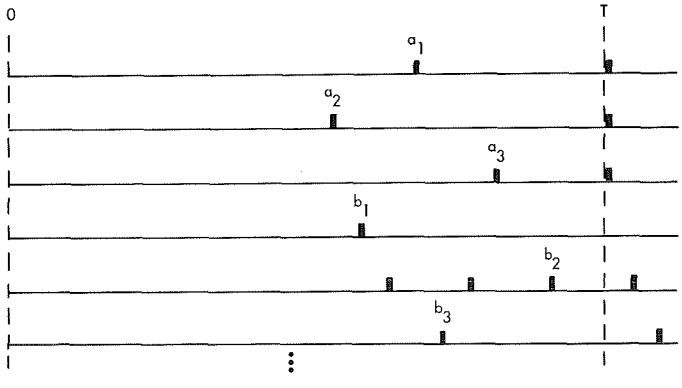


Fig. 18. Processing overflow at time T

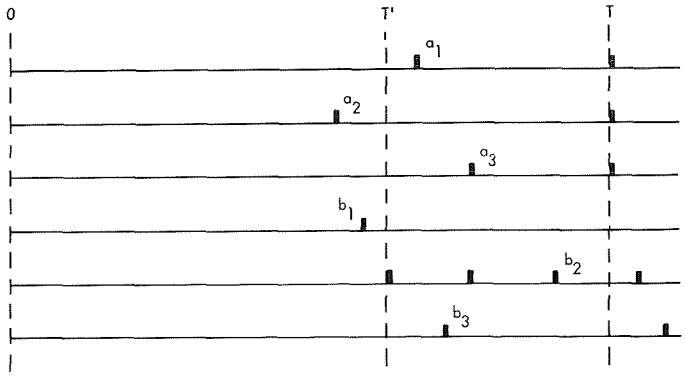


Fig. 19. Processing overflow at time T without execution of $\{b_i\}$ following T'

Since

$$\frac{T}{T_1} > \left[\frac{T}{T_1}\right], \frac{T}{T_2} \geq \left[\frac{T}{T_2}\right], \cdots, \frac{T}{T_m} \geq \left[\frac{T}{T_m}\right]$$

we have

$$\frac{T}{T_1}C_1 + \frac{T}{T_2}C_2 + \cdots + \frac{T}{T_m}C_m > T$$

or

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_m}{T_m} > 1$$

which is a contradiction to Inequality (1).

Case 2. Some of the computations requested at $b_1, b_2, \cdots$ were carried out before T. Since an overflow occurs at T, there must exist a point T' such that none of the requests at $b_1, b_2, \cdots$ is carried out within the interval $T' \leq t \leq T$. In other words, within $T' \leq t \leq T$, only those requests with deadlines at or before T will be executed, as illustrated in Fig. 19. Moreover, the fact that one or more of the tasks having requests at the b_i's is executed until $t = T'$ means that all those requests initiated before T' with deadlines at or before T have been fulfilled before T'. Therefore, the total demand of processor time within $T' \leq t \leq T$ is less than or equal to

$$\left[\frac{T - T'}{T_1}\right]C_1 + \left[\frac{T - T'}{T_2}\right]C_2 + \cdots + \left[\frac{T - T'}{T_m}\right]C_m$$

That an overflow occurs at T means that

$$\left[\frac{T-T'}{T_1}\right]C_1+\left[\frac{T-T'}{T_2}\right]C_2+\cdots+\left[\frac{T-T'}{T_m}\right]C_m>T-T'$$

which implies again

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\cdots+\frac{C_m}{T_m}>1$$

and we have a contradiction to Inequality (1). Theorem 1 is proved.

Corollary 1-1. The deadline-driven scheduling algorithm is optimum.

Proof. No scheduling algorithm at all is feasible for a set of tasks if

$$\frac{C_1}{T_1}+\frac{C_2}{T_1}+\cdots+\frac{C_m}{T_m}>1$$

which is the converse of the necessary and sufficient conditions for the deadline-driven algorithm to work.

c. Combination of period-driven and deadline-driven scheduling algorithm. We now derive the necessary and sufficient condition for the feasibility of a class of scheduling algorithms which are combinations of the period-driven scheduling algorithm and the deadline-driven scheduling algorithm. We call such a scheduling algorithm a *mixed* scheduling algorithm. The study of the mixed algorithms is motivated by the observation that the interrupt hardware of present-day computers acts as a fixed-priority scheduler. The deadline-driven scheduler could perhaps be best implemented as a software scheduler for the slower tasks. This implementation should be considerably cheaper than a hardware deadline-driven scheduler, and, as will be seen, will provide most of the advantages of one. To be specific, let tasks 1, 2, $\cdots$, k, the k tasks of shortest periods, be scheduled according to the period-driven scheduling algorithm, and let the remaining tasks, tasks $k+1$, $k+2$, $\cdots$, m, be scheduled according to the deadline-driven scheduling algorithm when the processor is not occupied by tasks 1, 2, $\cdots$, k.

Let $a(t)$ be a non-decreasing function of t; $a(t)$ is said to be sublinear if, for all t and all T,

$$a(T)\leq a(t+T)-a(t)$$

We define the availability function of a processor for a set of tasks as the accumulated processor time from 0 to t available to this set of tasks. Suppose that k tasks have been scheduled on a processor by the period-driven scheduling algorithm. We let $a_k(t)$ denote the availability function of the processor for tasks $k+1$, $k+2$, $\cdots$, m. Clearly, $a_k(t)$ is non-decreasing.

Lemma 2. $a_k(t)$ is sublinear.

Proof. Use the critical time zone argument as in SPS 37-59, Vol. II, pp. 48–54.

Lemma 3. If a set of tasks are scheduled by the deadline-driven scheduling algorithm on a processor whose availability function is sublinear, then there is no processor idle-period prior to an overflow.

Proof. Similar to that of Lemma 1.

Theorem 2. A necessary and sufficient condition for the feasibility of the deadline-driven scheduling algorithm with respect to a processor with availability function $a_k(t)$ is

$$\left[\frac{t}{T_{k+1}}\right]C_{k+1}+\left[\frac{t}{T_{k+2}}\right]C_{k+2}+\cdots+\left[\frac{t}{T_m}\right]C_m\leq a_k(t)$$

for all t's which are multiples of T_{k+1}, or of T_{k+2}, $\cdots$, or of T_m.

Proof. The proof is quite similar to that of Theorem 1. To show the necessity, we observe that at any moment, the total demand of processor time cannot exceed the total available processor time. Thus, we must have

$$\left[\frac{t}{T_{k+1}}\right]C_{k+1}+\left[\frac{t}{T_{k+2}}\right]C_{k+2}+\cdots+\left[\frac{t}{T_m}\right]C_m\leq a_k(t)$$

for all t.

To show the sufficiency, we assume that the condition stated in the theorem is satisfied and yet there is an overflow at T. We examine the two cases considered in the proof of Theorem 1. For case 1, we have the inequality

$$\left[\frac{T}{T_{k+1}}\right]C_{k+1}+\left[\frac{T}{T_{k+2}}\right]C_{k+2}+\cdots+\left[\frac{T}{T_m}\right]C_m>a_k(T)$$

which is a contradiction to our assumption. Note that T is multiple of T_{k+1}, or T_{k+2}, $\cdots$, or T_m. For case 2, we have the inequality

$$\left[\frac{T-T'}{T_{k+1}}\right]C_{k+1}+\left[\frac{T}{T_{k+1}}\right]C_{k+2}+\cdots+\left[\frac{T-T'}{T_m}\right]C_m > a_k(T-T')$$

Let ϵ be the smallest non-negative quantity such that $T-T'-\epsilon$ is a multiple of T_{k-1}, or T_{k-2}, $\cdots$, or T_m. We have

$$\left[\frac{T-T'-\epsilon}{T_{k-1}}\right]=\left[\frac{T-T'}{T_{k-1}}\right],\left[\frac{T-T'-\epsilon}{T_{k-2}}\right]=\left[\frac{T-T'}{T_{k+2}}\right],\cdots,\left[\frac{T-T'-\epsilon}{T_m}\right]=\left[\frac{T-T'}{T_m}\right]$$

and thus

$$\left[\frac{T-T'-\epsilon}{T_{k+1}}\right]C_{k+1}+\left[\frac{T-T'-\epsilon}{T_{k+2}}\right]C_{k+2}+\cdots+\left[\frac{T-T'-\epsilon}{T_m}\right]C_m > a_k(T-T') \text{ and } \geq a_k(T-T'-\epsilon)$$

which is a contradiction to our assumption.

d. A special case of the mixed scheduling algorithm. Although the result in Theorem 2 is a useful general result, its application involves the solution of a large set of inequalities. We investigate now a special case in which three tasks are scheduled by the mixed scheduling algorithm such that the task with the shortest period is scheduled by the period-driven scheduling and the other two tasks are scheduled by the deadline-driven scheduling algorithm. We want to illustrate how sufficient conditions on feasibility can be derived from the result in Theorem 2. We have:

Theorem 3. If

$$1-\frac{C_1}{T_1}-\min\left(\frac{T_1-C_1}{T_2},\frac{C_1}{T_2}\right)\geq\frac{C_2}{T_2}+\frac{C_3}{T_3}$$

then the mixed scheduling algorithm is feasible.

Proof. According to Theorem 2, a necessary and sufficient condition for the mixed scheduling algorithm to be feasible is

$$\left[\frac{t}{T_2}\right]C_2+\left[\frac{t}{T_3}\right]C_3\leq f(t,T_1,C_1) \qquad (2)$$

for all t's which are multiples of T_2 or T_3;$f(t,A,B)$ denotes

$$\left[\frac{t}{A}\right](A-B)+\max\left(o,\left\{\frac{t}{A}\right\}-B\right)$$

$\{x\}$ denotes $x-[x]$, the fractional part of x.

Since

$$f(t,T_1,C_1)\geq\left[\frac{t}{T_1}\right](T_1-C_1)+\left\{\frac{t}{T_1}\right\}-C_1$$

and

$$f(t,T_1,C_1)\geq\left[\frac{t}{T_1}\right](T_1-C_1)$$

Inequality (2) is implied either by

$$\left[\frac{t}{T_2}\right]C_2+\left[\frac{t}{T_3}\right]C_3\leq\left[\frac{t}{T_1}\right](T_1-C_1)+\left\{\frac{t}{T_1}\right\}-C_1 \qquad (3)$$

or by

$$\left[\frac{t}{t_2}\right]C_2+\left[\frac{t}{T_3}\right]C_3\leq\left[\frac{t}{T_1}\right](T_1-C) \qquad (4)$$

Since

$$\begin{aligned}\left[\frac{t}{T_1}\right](T_1-C_1)+\left\{\frac{t}{T_1}\right\}-C_1&=\left[\frac{t}{T_1}\right]T_1-\left[\frac{t}{T_1}\right]C_1+t-\left[\frac{t}{T_1}\right]T_1-C_1\\&=t-\left[\frac{t}{T_1}\right]C_1-C_1\\&\geq t-\frac{t}{T_1}C_1-C_1\end{aligned}$$

Inequality (3) is implied by

$$\left[\frac{t}{T_2}\right] C_2 + \left[\frac{t}{T_3}\right] C_3 \leq t - \frac{t}{T_1} C_1 - C_1$$

which, in turn, is implied by

$$\frac{t}{T_2} C_2 + \frac{t}{T_3} C_3 \leq t - \frac{t}{T_1} C_1 - C_1$$

which can be written as

$$\frac{C_2}{T_2} + \frac{C_3}{T_3} \leq 1 - \frac{C_1}{T_1} - \frac{C_1}{t}$$

Since

$$1 - \frac{C_1}{T_1} - \frac{C_1}{t} \geq 1 - \frac{C_1}{T_1} - \frac{C_1}{T_2}$$

for all t's which are multiples of T_2 or T_3, we conclude that Inequality (3) is implied by the inequality

$$\frac{C_2}{T_2} + \frac{C_3}{T_3} \leq 1 - \frac{C_1}{T_1} - \frac{C_1}{T_2}$$

Since

$$\left[\frac{t}{T_1}\right] (T_1 - C_1) \geq \left(\frac{t}{T_1} - 1\right) (T_1 - C_1)$$

Inequality (4) is implied by

$$\left[\frac{t}{T_2}\right] C_2 + \left[\frac{t}{T_3}\right] C_3 \leq \left(\frac{t}{T_1} - 1\right) (T_1 - C_1)$$

which is, in turn, implied by

$$\frac{t}{T_2} C_2 + \frac{t}{T_3} C_3 \leq \frac{t}{T_1} (T_1 - C_1) - (T_1 - C_1)$$

which can be written

$$\frac{C_2}{T_2} + \frac{C_3}{T_3} \leq \frac{T_1 - C_1}{T_1} - \frac{T_1 - C_1}{t} \tag{5}$$

Again, Inequality (5) is implied by

$$\frac{C_2}{T_2} + \frac{C_3}{T_3} \leq 1 - \frac{C_1}{T_1} - \frac{T_1 - C_1}{T_2}$$

which proves the result. We also have:

Theorem 4. If the following three inequalities hold,

$$C_2 \leq a_1(T_2)$$

$$\left[\frac{T_3}{T_2}\right] C_2 + C_3 \leq a_1\left(\left[\frac{T_3}{T_2}\right] T_2\right)$$

$$\left(\left[\frac{T_3}{T_2}\right] + 1\right) C_2 + C_3 \leq a_1(T_3)$$

then the mixed scheduling algorithm is feasible.

Proof. To simplify the notations, let $k = [T_3/T_2]$. According to Theorem 2, the following two inequalities must be satisfied:

$$iC_2 + \left[\frac{iT_2}{T_3}\right] C_3 \leq a_1(iT_2), \qquad i = 1, 2, \cdots \tag{6}$$

$$\left[\frac{iT_3}{T_2}\right] C_2 + iC_3 \leq a_1(iT_3), \qquad i = 1, 2, \cdots \tag{7}$$

Since $j\, a_1(T_2) \leq a_1(jT_2)$ for any $j \geq 0$, the inequality

$$C_2 \leq a_1(T_2) \tag{8}$$

implies all inequalities of the form

$$jC_2 \leq a_1(jT_2), \qquad j \geq 0$$

We now claim that Inequality (8), together with

$$kC_2 + C_3 \leq a_1(kT_2) \tag{9}$$

implies all inequalities in Inequality (6). Let

$$\left[\frac{iT_2}{T_3} = p\right]$$

Then $i > pk$. Let $i = pk + d$, $d > 0$. Addition of Inequalities (8) and (9) weighted by d and p, respectively, gives

$$\begin{aligned}(pk + d) C_2 + pC_3 &\leq p\, a_1(kT_2) + d\, a_1(T_2) \\ &\leq a_1(pkT_2) + a_1(dT_2) \\ &\leq a_1((pk + d) T_2)\end{aligned}$$

That is,

$$i\,C_2 + \left\lceil \frac{iT_2}{T_3} \right\rceil C_3 \leq a_1(iT_2)$$

We show next that the inequality

$$(k+1)\,C_2 + C_3 \leq a_1(T_3)$$

implies all inequalities in Inequality (2). Since $i(k+1) \geq [iT_2/T_2]$, the inequality

$$i(k+1)\,C_2 + i\,C_3 \leq i\,a_1(T_3)$$

implies

$$\left\lceil \frac{iT_3}{T_2} \right\rceil C_2 + i\,C_3 \leq i\,a_1(T_3)$$

which in turn implies

$$\left\lceil \frac{iT_3}{T_2} \right\rceil C_2 + i\,C_3 \leq a_1(iT_3)$$

The theorem is proved.

e. Comparison and comment. The constraints developed by Theorems 2 to 4 strongly suggest that 100% utilization is not achievable universally by the mixed scheduling algorithm. The following simple example will illustrate: Let $T_1 = 3$, $T_2 = 4$, $T_3 = 5$, and $C_1 = C_2 = 1$. Since $a_1(20) = 13$, it can be easily seen that the maximum allowable $C_3 = 2$. The corresponding utilization factor is

$$\mu = \frac{1}{3} + \frac{1}{4} + \frac{2}{5} = 98.3\%$$

If these three tasks are scheduled by the deadline scheduling algorithm, C_2 can increase to $2.0833\cdots$ and achieve 100% utilization. If they are all scheduled by the period-driven scheduling algorithm, C_2 is restricted to 1 or less and utilization is restricted to at most

$$\mu = \frac{1}{3} + \frac{1}{4} + \frac{1}{5} = 78.3\%$$

which is only slightly greater than the worst-case three-task utilization bound.

Although a closed-form expression for the least upper bound to processor utilization has not been found for the mixed scheduling algorithm, this example strongly suggests that the bound is considerably less restrictive for the mixed algorithm than for the period-driven algorithm. The mixed algorithm may thus be appropriate for many applications, since it can be implemented via the interrupt hardware of present day computers, and provides most of the capability of the fully deadline-driven algorithm.

B. Communications Elements Research

1. Improved RF Calibration Techniques: System-Operating Noise-Temperature Calibrations of the JPL Research Cones, C. T. Stelzried

The system-operating noise-temperature performance of the low-noise research cones is reported. The operating noise-temperature calibrations are performed with the ambient termination technique (SPS 37-42, Vol. III, pp. 25–32). The principal advantage of this method is the stability and reliability of the ambient termination.

The research cones using this operational technique during this reporting period are:

(1) S-band planetary radar (SPR) cone.

(2) S-band research operational (SRO) cone.

(3) S-band cassegrain ultra (SCU) cone.

These cones are operated at the DSS 13 and 14 antennas in a cassegrain configuration.[1] The calibration parameters of the JPL research cones are summarized in Table 7.

The averaged operating noise-temperature calibrations were taken for the various research cone configurations during the period June 1 through October 1, 1969. The various cases are defined in Table 8. The calibrations are presented in Table 9. The data taken at DSS 14 with maser 2 (cases 7 and 8) used an aperture load placed in position by hand over the horn opening.

The operating noise-temperature data were reduced with JPL computer program ID 5841000, CTS 20B. The indicated errors in Table 9 are the standard deviation of the individual measurements and of the means, and

[1]The calibration data were taken by Goldstone DSCC personnel at the stations indicated.